ServoCenter 4.1

Volume 3:

Programming Guide
&
Programming Examples

Yost Engineering, Inc.
630 Second Street
Portsmouth, Ohio 45662

www.YostEngineering.com

©2002-2009 Yost Engineering, Inc.
Printed in USA

Table of Contents

1. Programming with Raw Serial I/O.....3
1.1 QBASIC Example Program.....3
1.2 C++: Microsoft Visual C++ 6 Example Program.....4
1.3 C : Linux GCC Example Program.....6
1.4 C : Borland Turbo C Sample Program.....8
1.5 Visual Basic 6 Sample Program.....9
1.6 Python Sample Program.....10
2. Programming With the ServoCenter 4.1 ActiveX Control.....11
2.1 Operation with ServoCenter 4.1 ActiveX Control.....11
2.2 Installing the ServoCenter 4.1 ActiveX Control.....11
2.3 Using the ServoControl in Visual Basic 6.0.....12
2.4 ServoCenter 4.1 OCX Control Methods.....13
Public Properties.....13
Port Functions.....13
Normal Movement Group Commands.....15
Compact Movement Servo Commands.....16
Compact Movement Group Commands.....19
Set Servo Settings Commands.....20
Get Servo Settings Commands.....21
Input/Output Commands.....22
Servo Group Mask Commands.....23
Preset Commands.....24
Sequencer / BASIC Interpreter Commands.....26
General Commands.....27
2.5 Programming in Visual Basic 6.0 with the YEIServoControl.....27
3. Programming With the ServoCenter 4.1 DLL.....28
3.1 ServoCenter 4.1 DLL Functional Overview.....28
Port Functions.....28
Normal Movement Servo Commands.....28
Normal Movement Group Commands.....31
Compact Movement Servo Commands.....32
Compact Movement Group Commands.....35
Set Servo Settings Commands.....37
Get Servo Settings Commands.....39
Input/Output Commands.....40
Servo Group Mask Commands.....41
Preset Commands.....43
Sequencer / BASIC Interpreter Commands.....45
General Commands.....46
3.2 Installing the yeisrvo.dll Runtime Library.....47
3.3 Programming with yeisrvo.dll in Visual Basic 6.0.....47
3.4 Programming with yeisrvo.dll in Visual C++ 6.0.....49
3.5 Programming with yeisrvo.dll in Visual C++ .NET.....50
3.6 Programming with yeisrvo.dll in the Microsoft .NET Framework.....50
Visual Basic .NET.....51
C#.....51
4. Programming With the ServoCenter 4.1 .NET Assembly.....52
4.1 Operation with ServoCenter 4.1 .NET Assembly.....52
4.2 Installing the ServoCenter 4.1 .NET Assembly.....52
4.3 Using the ServoCenter 4.1 .NET Assembly in Visual Basic 2008.....52
4.4 ServoCenter 4.1 .NET Assembly Methods.....53
5. Appendix.....54
5.1 Hexadecimal/Decimal/Binary Conversion Chart.....54
5.2 Serial Cable Diagram.....54
5.3 ServoCenter 4.1 Circuit Schematic.....55

1. Programming with Raw Serial I/O

The following section provides simple example programs in a variety of programming languages and environments. Each example program illustrates how to access the serial port and directly communicate with the ServoCenter 4.1 controller board to control a servo. Note that the programs are provided to illustrate simple raw serial communication using the ServoCenter 4.1 protocol and do not demonstrate the full feature set of the ServoCenter 4.1 controller. Refer to the document: "Volume 2: Protocol Reference" for a description of the entire ServoCenter 4.1 protocol and feature set.

1.1 QBASIC Example Program

```
DECLARE SUB initCOMPort (port AS INTEGER,baud as INTEGER)
DECLARE SUB MoveServoRaw (BoardNum AS INTEGER, ServoNum AS INTEGER,_
     Position AS INTEGER, Speed AS INTEGER)

    '*************************************************************
    '*  This demo program illustrates how to move servo motors   *
    '*  using raw serial communication access to the             *
    '*  Yost Engineering, Inc. ServoCenter 4.1 controller board. *
    '*                                                           *
    '*         (c) 2001-2010   Yost Engineering, Inc.            *
    '*                www.YostEngineering.com                    *
    '*                                                           *
    '*************************************************************

    DIM svBoardnum AS INTEGER
    DIM svServonum AS INTEGER
    DIM svPosition AS INTEGER

    CLS
    COLOR 15, 1
    PRINT "                                                      "
    PRINT "        ServoCenter 4.1 Demonstration Program         "
    PRINT "        (c)2000-2010 Yost Engineering, Inc.           "
    PRINT "             www.YostEngineering.com                  "
    PRINT "                                                      "
    PRINT " This program demonstrates the QuickMove Raw command.  "
    PRINT "                                                      "
    COLOR 7, 0

    PRINT ""
    'initialize the serial port com1 to 9600
    CALL initCOMPort(1,9600)

    DO WHILE 1 = 1

       INPUT "   Enter a board number (0-15): ", svBoardnum
       INPUT "   Enter a servo number (0-15): ", svServonum
       INPUT "   Enter a position value (0-16383): ", svPosition
       COLOR 4, 0
       PRINT "     Sending QuickMove Raw Command now...       "
       CALL MoveServoRaw(svBoardnum, svServonum, svPosition)
       COLOR 2, 0
       PRINT "     Sent!": PRINT
       COLOR 7, 0
    LOOP

SUB initCOMPort (port AS INTEGER, baud as INTEGER)
    settings$ = "COM"+LTRIM$(STR$(port))+":"+LTRIM$(STR$(baud))+_
      ",N,8,1,CD0,CS0,DS0"
    OPEN settings$ FOR RANDOM AS #1
END SUB
```

```
SUB MoveServoRaw (BoardNum AS INTEGER, ServoNum AS INTEGER, _
     Position AS INTEGER)
    PRINT #1, CHR$(&HF0 + BoardNum MOD 16); CHR$(0);
    PRINT #1, CHR$(ServoNum MOD 16);
    PRINT #1, CHR$(int(Position / 128)); CHR$(Position MOD 128);
    PRINT #1, CHR$(0);
END SUB
```

1.2 C++: Microsoft Visual C++ 6 Example Program

```
/*********************************************************\
*  This demo program illustrates how to move servo motors   *
*  using raw serial communication access to the             *
*  Yost Engineering, Inc. ServoCenter 4.1 controller board  *
*  using Visual C++ 6.0.                                    *
*                                                           *
*         (c) 2001-2010   Yost Engineering, Inc.            *
*                www.YostEngineering.com                    *
*                                                           *
\*********************************************************/

#include <windows.h>
#include <stdio.h>
#define PORTNUM 1
#define BAUDRATE 9600

void moveservo(HANDLE *,int,int,int);
int InitPort(unsigned int, unsigned int, HANDLE *, DCB *);

int main(int argc, char *argv[])
{
   DCB dcb;
   HANDLE hCom;
   int i=0, BoardNum, ServoNum, Position;

   printf("                                                    \n");
   printf("       ServoCenter 4.1 Demonstration Program        \n");
   printf("        (c)2000-2010 Yost Engineering, Inc.         \n");
   printf("             www.YostEngineering.com                \n");
   printf("                                                    \n");
   printf(" This program demonstrates the QuickMove Raw command.\n");
   printf("                                                    \n");

   if((InitPort(PORTNUM,BAUDRATE,&hCom,&dcb))!=0)//open serial port
   {
        printf("\tCould not initialize Comm Port!\n");
        return (1);
   }
   else
   {
        while(1)
        {
               printf("\n  Enter Board Number (0-15):");
               scanf("%d",&BoardNum);
               printf("\n  Enter Servo Number (0-15):");
               scanf("%d",&ServoNum);
               printf("\n  Enter Position (0-16383):");
               scanf("%d",&Position);
               printf("\n\tSending QuickMove Raw Command now...\n");
               moveservo(&hCom,BoardNum,ServoNum,Position);
               printf("\n\tDone!\n");
        }
   }
   return (0);
}

void moveservo(HANDLE *hCom,int board,int servo,int position)
{
  unsigned char buffer[6];      //create empty command packet
```

```
  unsigned long BytesWritted;   //records # of bytes sent

  buffer[0]=board%16 + 0xf0;    //board id #
  buffer[1]=0;             //QuickMove Raw command
  buffer[2]=servo%16;      //servo #--used to identify which servo
  buffer[3]=position/128;  //position MSB 7 bits
  buffer[4]=position%128; //position LSB 7 bits
  buffer[5]='\0';         //NULL character added to disable checksum
  WriteFile(*hCom,buffer,5,&BytesWritted,NULL); //send packet
}



int InitPort(unsigned int PortNum, unsigned int BaudRate, HANDLE *hCom,
DCB *dcb)
{
     BOOL fSuccess;
     char pcCommPort[4]={'\0'};
     if(PortNum==1)
          sprintf(pcCommPort,"COM1");
     else if(PortNum==2)
          sprintf(pcCommPort,"COM2");
     else if(PortNum==3)
          sprintf(pcCommPort,"COM3");
     else if(PortNum==4)
          sprintf(pcCommPort,"COM4");
     else
          printf("\tPort Number not recognized\n");

     *hCom = CreateFile( pcCommPort,
                    GENERIC_READ | GENERIC_WRITE,
                    0,    // must be opened with exclusive-access
                    NULL, // no security attributes
                    OPEN_EXISTING, // must use OPEN_EXISTING
                    0,    // not overlapped I/O
                    NULL  // hTemplate must be NULL for comm
                    );

   if (*hCom == INVALID_HANDLE_VALUE)
   {
       // Handle the error.
       printf("\tCreateFile failed with error%d.\n",
           GetLastError());
       return (2);
   }


   // Build on the current configuration, and skip setting the size
   // of the input and output buffers with SetupComm.

   fSuccess = GetCommState(*hCom, dcb);

   if (!fSuccess)
   {
      // Handle the error.
      printf ("\tGetCommState failed with error %d.\n",
           GetLastError());
      return (3);
   }

      // Fill in DCB: Baudrate,8 data bits,no parity, 1 stop bit.
   switch(BaudRate)
     {
          case 9600:  dcb->BaudRate = CBR_9600; break;
          case 38400: dcb->BaudRate = CBR_38400; break;
          case 57600: dcb->BaudRate = CBR_57600; break;
          case 115200: dcb->BaudRate = CBR_115200; break;
          default:  printf("\tBaud Rate not recognized\n");
                    printf("\tUsing default rate of 9600bps.\n");
                    dcb->BaudRate = CBR_9600; break;
     }
```

```
   dcb->ByteSize = 8;             // data size, xmit, and rcv
   dcb->Parity = NOPARITY;        // no parity bit
   dcb->StopBits = ONESTOPBIT;    // one stop bit

   fSuccess = SetCommState(*hCom, dcb);

   if (!fSuccess)
   {
      // Handle the error.
      printf("\tSetCommState failed with error %d.\n",
           GetLastError());
      return (4);
   }

   printf ("\tSerial port successfully reconfigured.\n");
   return (0);
}
```

1.3 C : Linux GCC Example Program

```
/*********************************************************\
*  This demo program illustrates how to move servo motors   *
*  using raw serial communication access to the             *
*  Yost Engineering, Inc. ServoCenter 4.1 controller board. *
*  in Linux.  This code was compiled using GCC.             *
*         (c) 2001-2010   Yost Engineering, Inc.            *
*               www.YostEngineering.com                     *
*                                                           *
\*********************************************************/

#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>

int open_port(int portnum)
{
  int fd;
  char portfile[100]={'\0'};
  if(portnum==1)
    sprintf(portfile,"/dev/ttyS0");
  else if(portnum==2)
    sprintf(portfile,"/dev/ttyS1");
  else if(portnum==3)
    sprintf(portfile,"/dev/ttyS2");
  else if(portnum==4)
    sprintf(portfile,"/dev/ttyS3");
  else
  {
    printf("open_port: unrecognized port number\n");
    return (-1);
  }
  if((fd=open(portfile, O_RDWR | O_NOCTTY | O_NDELAY))==-1)
    perror("open_port: unable to open /dev/ttyS0 - ");
  return (fd);
}

void init_port(int *fd, unsigned long baud)
{
    struct termios options;
    //note: the termios structure does not support a baud rate of 14400
    tcgetattr(*fd,&options);
```

```
    switch(baud)
    {
       case 9600:    cfsetispeed(&options,B9600);
                     cfsetospeed(&options,B9600);
                     break;
       case 38400:   cfsetispeed(&options,B38400);
                     cfsetospeed(&options,B38400);
                     break;
       case 57600:   cfsetispeed(&options,B57600);
                     cfsetospeed(&options,B57600);
                     break;
       case 115200:  cfsetispeed(&options,B115200);
                     cfsetospeed(&options,B115200);
                     break;
        default:     cfsetispeed(&options,B9600);
                     cfsetospeed(&options,B9600);
                     break;
    }
    options.c_cflag |= (CLOCAL | CREAD);
    options.c_cflag &= ~PARENB;
    options.c_cflag &= ~CSTOPB;
    options.c_cflag &= ~CSIZE;
    options.c_cflag |= CS8;
    tcsetattr(*fd,TCSANOW,&options);
}

void moveservo(int *fd, int boardnum, int servonum, int position)
{
    char buffer[6];
    int num;
    buffer[0]=boardnum%16 + 0xf0; // Board ID
    buffer[1]=0x00;              //command ID for QuickMove Raw
    buffer[2]=servonum%16;       // servo number
    buffer[3]=position/128;      // raw position MSB 7-bits
    buffer[4]=position%128;      // raw position LSB 7-bits
    buffer[5]='\0';              // checksum of 0 is ignored.
    num=write(*fd,buffer,5);     //send packet
}

int main()
{
  int fd,board,servo,position,portnum;
  printf("                                                    \n");
  printf("        ServoCenter 4.1 Demonstration Program       \n");
  printf("        (c)2000-2010 Yost Engineering, Inc.         \n");
  printf("             www.YostEngineering.com                \n");
  printf("                                                    \n");
  printf(" This program demonstrates the QuickMove Raw command.\n");
  printf("                                                    \n");
  printf("Enter Port Number (1-4)\n");
  scanf("%d",&portnum);
  if((fd=open_port(portnum))==-1)   //open serial port
     return (1);
  init_port(&fd,9600);         //set serial port to 9600,8,n,1
  while(1)
  {
    printf("Enter Board Number (0-15)\n");
    scanf("%d",&board);
    printf("Enter Servo Number (0-15)\n");
    scanf("%d",&servo);
    printf("Enter Position (0-16383)\n");
    scanf("%d",&position);
    printf("Sending Command...");
    moveservo(&fd,board,servo,position);
    printf("done!\n");
  }
  return (0);
}
```

1.4 C : Borland Turbo C Sample Program

```
/*********************************************************\
*  This demo program illustrates how to move servo motors   *
*  using raw serial communication access to the             *
*  Yost Engineering, Inc. ServoCenter 4.1 controller board. *
*  This program was written and compiled in the Borland     *
*  Turbo C environment.                                     *
*                                                           *
*         (c) 2001-2010   Yost Engineering, Inc.            *
*                www.YostEngineering.com                    *
*                                                           *
\*********************************************************/

#include<stdio.h>
#include<dos.h>
#include<conio.h>

#define COM1 0x3f8
#define COM2 0x2f8
#define COM3 0x3e8
#define COM4 0x2e8
/* Set following line to desired port*/
#define COMPORT COM1
#define BAUDRATE 9600

void initcom(unsigned long);
void moveservo(int,int,int);

main()
{
  int board,servo,pos;
  clrscr();
  printf("                                                    \n");
  printf("        ServoCenter 4.1 Demonstration Program        \n");
  printf("        (c)2000-2010 Yost Engineering, Inc.         \n");
  printf("             www.YostEngineering.com                \n");
  printf("                                                    \n");
  initcom(BAUDRATE);
  while (1)
  {
      printf("Enter Board ID (0-15)\n");
      scanf("%d",&board);
      printf("Enter ServoID (0-15)\n");
      scanf("%d",&servo);
      printf("Enter Position (0-16383)\n");
      scanf("%d",&pos);
      printf("Moving servo...\n");
      moveservo(board,servo,pos,speed);
      printf("Done!");
  }
}

void moveservo(int BoardId,int ServoNum,int Position)
{
  outportb(COMPORT,0xf0 + BoardId % 16);   // Board ID
  while((inportb(COMPORT+5)&0x20)==0){;}
  outportb(COMPORT,0x00);                  // Command ID for QuickMove Raw
  while((inportb(COMPORT+5)&0x20)==0){;}
  outportb(COMPORT,ServoNum % 16);         // Servo Number
  while((inportb(COMPORT+5)&0x20)==0){;}
  outportb(COMPORT,Position / 128);        // Position MSB 7-bits
  while((inportb(COMPORT+5)&0x20)==0){;}
  outportb(COMPORT,Speed % 128);           // Position LSB 7-bits
  while((inportb(COMPORT+5)&0x20)==0){;}
  outportb(COMPORT,0);                     // Checksum of 0 is ignored.
  while((inportb(COMPORT+5)&0x20)==0){;}
}
```

```
void initcom( unsigned long BaudRate )
{
      outportb(COMPORT+3,0x83); //DLAB high, set format 8N1
      switch(BaudRate)
      {
            case 9600: outportb(COMPORT,0x0c);   // set rate LSB
                        outportb(COMPORT+1,0x00); // set rate MSB
                        break;
            case 38400: outportb(COMPORT,0x03);   // set rate LSB
                        outportb(COMPORT+1,0x00); // set rate MSB
                        break;
            case 57600: outportb(COMPORT,0x02);   // set rate LSB
                        outportb(COMPORT+1,0x00); // set rate MSB
                        break;
            case 115200:outportb(COMPORT,0x01);   // set rate LSB
                        outportb(COMPORT+1,0x00); // set rate MSB
                        break;
            default:   //use 9600 as default baud rate
                        outportb(COMPORT,0x0c);   // set rate LSB
                        outportb(COMPORT+1,0x00); // set rate MSB
                        break;
      }
      outportb(COMPORT+3,0x03); // DLAB now low
      outportb(COMPORT+1,0x00); // Interrupts off
}
```

1.5 Visual Basic 6 Sample Program

For the purpose of this sample, a simple VB form consisting of four Text Boxes, a Command Button, and the MSComm Control was used. Please refer to the project file on the CDROM for further details.

This example makes use of two program events: the Form_Load event and the cmdMove_Click event. The code attached to these events can be seen below.

```
' This program communicates with the ServoCenter 4.1 board
' by using the raw packet format in Visual Basic 6.  The serial
' port is accessed in this example via the Microsoft Comm Control
'
' (c) 2000-2010 Yost Engineering
'  www.YostEngineering.com

Private Sub cmdMove_Click()
    txtBoardNum.Text = Trim$(txtBoardNum.Text)
    txtServoNum.Text = Trim$(txtServoNum.Text)
    txtPosition.Text = Trim$(txtPosition.Text)
    MSComm1.Output = Chr$(&HF0 + Val(txtBoardNum.Text)) & Chr$(0)_
    & Chr$(Val(txtServoNum.Text)) & Chr$(int(Val(txtPosition.Text)/128))_
    & Chr$(int(Val(txtPosition.Text) Mod 128)) & Chr$(0)
End Sub

Private Sub Form_Load()
    MSComm1.CommPort = 1      'use comm1
    MSComm1.Settings = "9600,N,8,1"   'set up comm port.
    MSComm1.PortOpen = True   'open the port
End Sub
```

1.6 Python Sample Program

The Python sample program should run on any platform running Python 2.2 or newer so long as the pyserial Python module has been downloaded and properly installed. For installation on Windows systems, pywin32 extensions may also need to be installed.

```
# This program communicates with the ServoCenter 4.1 board
# by using the pyserial module.
#
# The program demonstrates using the Quickmove Raw command
# to move a servo.
#
# (c) 2000-2010 Yost Engineering
# www.YostEngineering.com
#

import serial

def quickMoveServoRaw(boardID,svNum, svPosition):
     outString = chr(0xf0+boardID)   # boardID byte
     outString += chr(0)             # command ID
     outString += chr(svNum)         # servo number
     outString += chr(int(svPosition/128)) # position MSB 7-bits
     outString += chr(svPosition%128) # position LSB 7-bits
     outString += chr(0)             # checksum byte
     serial_port.write( outString )

#open the port
serial_port = serial.Serial(port='COM1',baudrate=9600)

print "Port opened:",serial_port.portstr

# move the servo to the position specified.
boardID = int(raw_input("enter a board ID (0-15):"))
svNum = int(raw_input("enter a servo number (0-15):"))
svPosition = int(raw_input("enter a servo position (0-16383):"))


print "sending Quickmove Raw command..."

# send the message
quickMoveServoRaw(boardID,svNum,svPosition)

print "Done!"

# close the port
serial_port.close()
```

2. Programming With the ServoCenter 4.1 ActiveX Control

The ServoCenter 4.1 ActiveX Control can be used in a Visual Basic 6.0 project to control the ServoCenter 4.1. This control is designed to allow the programmer to communicate with the ServoCenter 4.1 board without having to worry about the communications protocol. Below is an explanation of how to control the ServoCenter 4.1 using the ServoCenter 4.1 ActiveX Control.

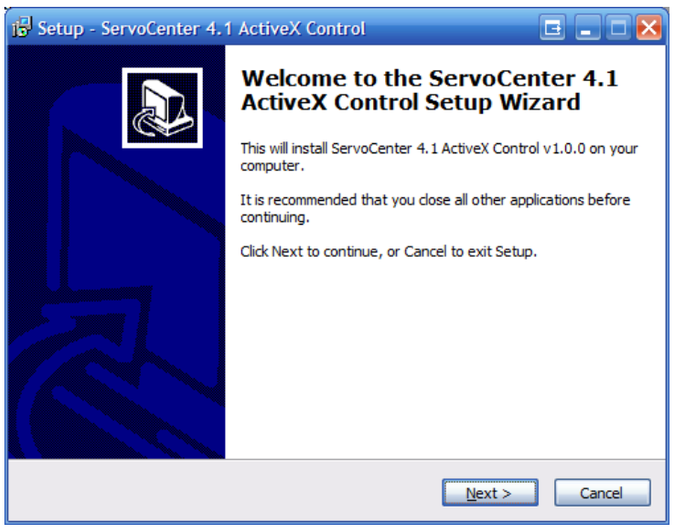
2.1 Operation with ServoCenter 4.1 ActiveX Control

The ServoCenter 4.1 CD contains the ServoCenter 4.1 ActiveX Control. The ServoCenter Control was designed to handle the details of raw serial communication with the ServoCenter 4.1.

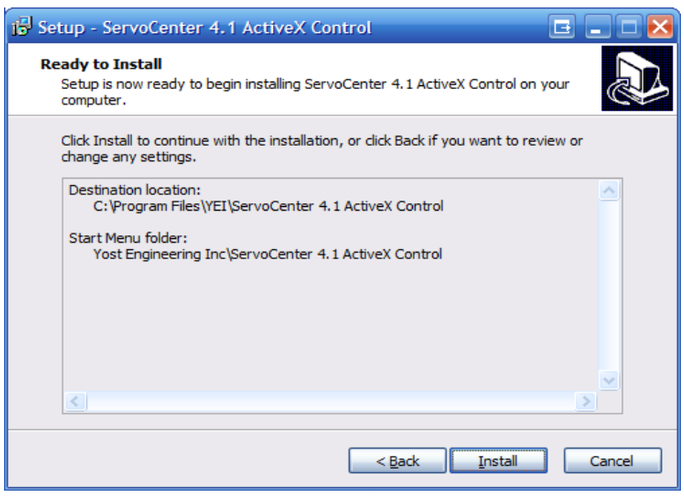
2.2 Installing the ServoCenter 4.1 ActiveX Control

1. Navigate to the “Resources” directory on the CD.

2. Run the ServoCenter 4.1 ActiveX Installer. To do this, double-click “ServoCenter 4.1 ActiveX Control Setup.exe” to begin the installation process.


Setup - ServoCenter 4.1 ActiveX Control
Welcome to the ServoCenter 4.1 ActiveX Control Setup Wizard
This will install ServoCenter 4.1 ActiveX Control v1.0.0 on your computer.
It is recommended that you close all other applications before continuing.
Click Next to continue, or Cancel to exit Setup.
Next >
Cancel

3. Follow the onscreen instructions to complete the installation.

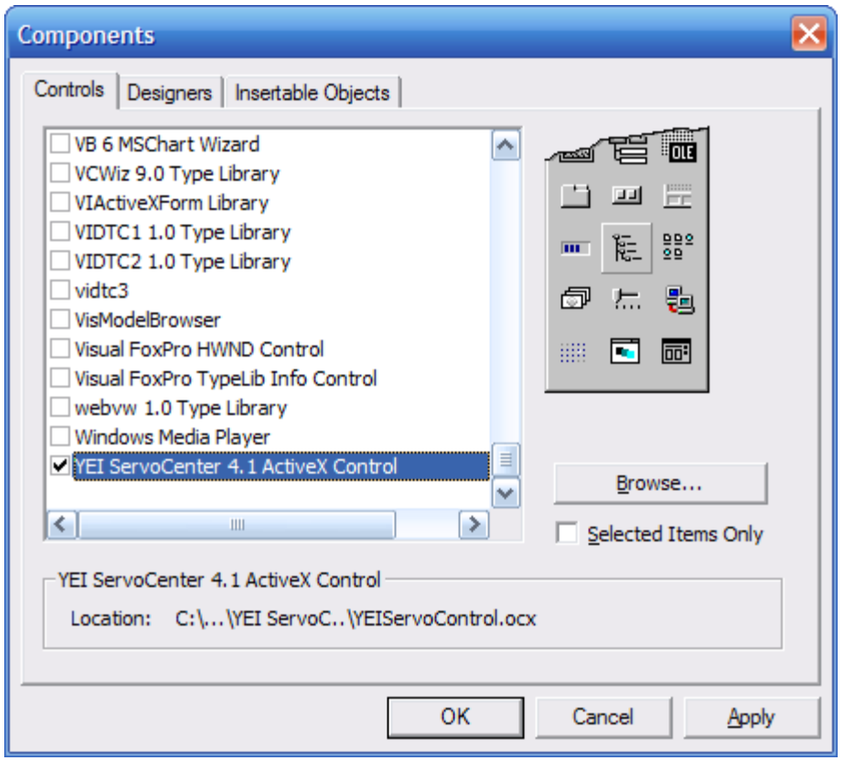

Setup - ServoCenter 4.1 ActiveX Control
Ready to Install
Setup is now ready to begin installing ServoCenter 4.1 ActiveX Control on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Destination location:
C:\Program Files\YEI\ServoCenter 4.1 ActiveX Control
Start Menu folder:
Yost Engineering Inc\ServoCenter 4.1 ActiveX Control
< Back
Install
Cancel

4. The control is now installed and ready for use!

2.3 Using the ServoControl in Visual Basic 6.0

1. Open a new project in the Visual Basic 6.0 editor.

2. Add the YEIServoControl to your project. To do this, click the “Project” menu at the top of the editor and then select “Components”. The “Components” menu will appear:



If the YEIServoControl is properly installed, it will appear in the dialog box. If not, it can still be added by clicking the “Browse” button and browsing to the directory where it is stored.

Once the YEIServoControl has been selected, click the “OK” button to add it to your project.

3. With the YEIServoControl now added to your project, select it from the toolbox and draw it on the form.

This completes the process of preparing the project for ServoCenter 4.1 use. You may now begin writing code to control the servos attached to your ServoCenter 4.1. To ensure the proper functionality of the YEIServoControl, please note that the **InitBoard** method must be invoked before attempting to access the ServoCenter 4.1 board. Once the InitBoard method has been executed, the servos attached to the ServoCenter 4.1 board can be manipulated.

2.4 ServoCenter 4.1 OCX Control Methods

Public Properties

BoardID – Selects which ServoCenter board will receive commands.

BaudRate – Data rate at which the com port will communicate with the board.

ComPort – Selects the com port to which the ServoCenter is connected.

Port Functions

Command:	InitBoard() As Boolean
Description:	None
Return Value:	True – Success False – Failure

Command:	GetAvailableComms() As Integer()
Description:	Used to query the system for a list of available com ports.
Return Value:	Array of integers containing available com ports on the system.

Command:	CloseCom()
Description:	Used to the close com port previously opened with the InitBoard() function.
Return Value:	None

Command:	ShowControlPanel()
Description:	Displays the integrated ServoCenter 4.1 Control Panel. The control panel can be used to configure and program the ServoCenter 4.1 board.
Return Value:	None

Command:	HideControlPanel()
Description:	Hides the ServoCenter 4.1 Control Panel.
Return Value:	None

Command:	QuickMoveRaw(ServoNumber As Integer, ServoPosition As Integer)
Parameters:	ServoNumber: ID (0~15) of servo whose position is to be changed. ServoPosition: Raw position (0~16383) to which servo will be moved.
Return Value:	None

Command:	QuickMoveScaled(ServoNumber As Integer, ServoPosition As Integer)
Parameters:	ServoNumber: ID (0~15) of servo whose position is to be changed. ServoPosition: Scaled position (0~16383) to which servo will be moved.
Return Value:	None

Command:	QuickMovePercent(ServoNumber As Integer, ServoPosition As Double)
Parameters:	ServoNumber: ID (0~15) of servo whose position is to be changed. ServoPosition: Scaled position (0~100) to which servo will be moved.
Return Value:	None

Command:	MoveRaw(ServoNumber As Integer, ServoPosition As Integer, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	MoveRawCW(ServoNumber As Integer, ServoPosition As Integer, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Units (0~16383) servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	MoveRawCCW(ServoNumber As Integer, ServoPosition As Integer, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Units (0~16383) servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	MoveScaled(ServoNumber As Integer, ServoPosition As Integer, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	MoveScaledCW(ServoNumber As Integer, ServoPosition As Integer, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	MoveScaledCCW(ServoNumber As Integer, ServoPosition As Integer, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	MovePercent(ServoNumber As Integer, ServoPosition As Double, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~100) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	MovePercentCW(ServoNumber As Integer, ServoPosition As Double, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~100) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	MovePercentCCW(ServoNumber As Integer, ServoPosition As Double, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~100) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	TimedMoveRaw(ServoNumber As Integer, ServoPosition As Integer, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Units (0~16383) servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	TimedMoveRawCW(ServoNumber As Integer, ServoPosition As Integer, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Units (0~16383) servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	TimedMoveRawCCW(ServoNumber As Integer, ServoPosition As Integer, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Units (0~16383) servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	TimedMoveScaled(ServoNumber As Integer, ServoPosition As Integer, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~16383) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	TimedMoveScaledCW(ServoNumber As Integer, ServoPosition As Integer, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~16383) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	TimedMoveScaledCCW(ServoNumber As Integer, ServoPosition As Integer, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~16383) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	TimedMovePercent(ServoNumber As Integer, ServoPosition As Double, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~100) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	TimedMovePercentCW(ServoNumber As Integer, ServoPosition As Double, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~100) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	TimedMovePercentCCW(ServoNumber As Integer, ServoPosition As Double, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~100) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Normal Movement Group Commands

Command:	GroupQuickMoveRaw(ServoPositions() As Integer)
Parameters:	ServoPositions: Position (0~16383) to which servos will be moved. Expects a 16 element array of integers. Values out of range are skipped to allow for the individual control of servos.
Return Value:	None

Command:	GroupQuickMoveScaled(ServoPositions() As Integer)
Parameters:	ServoPositions: Scaled position (0~16383) to which servos will be moved. Expects a 16 element array of integers. Values out of range are skipped to allow for the individual control of servos.
Return Value:	None

Command:	GroupQuickMovePercent(ServoPositions() As Double)
Parameters:	ServoPositions: Scaled position (0~100) to which servos will be moved. Expects a pointer to a 16 element array of doubles. Values out of range are skipped to allow for the individual control of servos.
Return Value:	None

Command:	GroupMoveRaw(ServoPositions() As Integer, ServoSpeed As Double)
Parameters:	ServoPositions: Position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. Values out of range are skipped to allow for the individual control of servos. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	GroupMoveScaled(ServoPositions() As Integer, ServoSpeed As Double)
Parameters:	ServoPositions: Scaled position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. Values out of range are skipped to allow for the individual control of servos. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	GroupMovePercent(ServoPositions() As Double, ServoSpeed As Double)
Parameters:	ServoPositions: Scaled position (0~100) to which servos will be moved. Expects a pointer to a 16 element array of floats. Values out of range are skipped to allow for the individual control of servos. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	GroupTimedMoveRaw(ServoPositions() As Integer, ServoTime As Double)
Parameters:	ServoPositions: Position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. Values out of range are skipped to allow for the individual control of servos. ServoTime: Length of time (0~23.9) in seconds for servo to move.
Return Value:	None

Command:	GroupTimedMoveScaled(ServoPositions() As Integer, ServoTime As Double)
Parameters:	ServoPositions: Position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. Values out of range are skipped to allow for the individual control of servos. ServoTime: Length of time (0~23.9) in seconds for servo to move.
Return Value:	None

Command:	GroupTimedMovePercent(ServoPositions() As Double, ServoTime As Double)
Parameters:	ServoPositions: Position (0~100) to which servos will be moved. Expects a pointer to a 16 element array of floats. Values out of range are skipped to allow for the individual control of servos. ServoTime: Length of time (0~23.9) in seconds for servo to move.
Return Value:	None

Compact Movement Servo Commands

Command:	CompactQuickMoveRaw(ServoNumber As Integer, ServoPosition As Integer)
Parameters:	ServoNumber: ID (0~15) of servo whose position is to be changed. ServoPosition: Raw position (0~16383) to which servo will be moved.
Return Value:	None

Command:	CompactQuickMoveScaled(ServoNumber As Integer, ServoPosition As Integer)
Parameters:	ServoNumber: ID (0~15) of servo whose position is to be changed. ServoPosition: Scaled position (0~16383) to which servo will be moved.
Return Value:	None

Command:	CompactQuickMovePercent(ServoNumber As Integer, ServoPosition As Double)
Parameters:	ServoNumber: ID (0~15) of servo whose position is to be changed. ServoPosition: Scaled position (0~100) to which servo will be moved.
Return Value:	None

Command:	CompactMoveRaw(ServoNumber As Integer, ServoPosition As Integer, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	CompactMoveRawCW(ServoNumber As Integer, ServoPosition As Integer, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Units (0~16383) servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	CompactMoveRawCCW(ServoNumber As Integer, ServoPosition As Integer, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Units (0~16383) servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	CompactMoveScaled(ServoNumber As Integer, ServoPosition As Integer, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	CompactMoveScaledCW(ServoNumber As Integer, ServoPosition As Integer, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	CompactMoveScaledCCW(ServoNumber As Integer, ServoPosition As Integer, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	CompactMovePercent(ServoNumber As Integer, ServoPosition As Double, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~100) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	CompactMovePercentCW(ServoNumber As Integer, ServoPosition As Double, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~100) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	CompactMovePercentCCW(ServoNumber As Integer, ServoPosition As Double, ServoSpeed As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~100) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	CompactTimedMoveRaw(ServoNumber As Integer, ServoPosition As Integer, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Units (0~16383) servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	CompactTimedMoveRawCW(ServoNumber As Integer, ServoPosition As Integer, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Units (0~16383) servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	CompactTimedMoveRawCCW(ServoNumber As Integer, ServoPosition As Integer, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Units (0~16383) servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	CompactTimedMoveScaled(ServoNumber As Integer, ServoPosition As Integer, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~16383) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	CompactTimedMoveScaledCW(ServoNumber As Integer, ServoPosition As Integer, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~16383) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	CompactTimedMoveScaledCCW(ServoNumber As Integer, ServoPosition As Integer, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~16383) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	CompactTimedMovePercent(ServoNumber As Integer, ServoPosition As Double, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~100) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	CompactTimedMovePercentCW(ServoNumber As Integer, ServoPosition As Double, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~100) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Command:	CompactTimedMovePercentCCW(ServoNumber As Integer, ServoPosition As Double, ServoTime As Double)
Parameters:	ServoNumber: ID (0~15) of servo to be moved. ServoPosition: Scaled position (0~100) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	None

Compact Movement Group Commands

Command:	CompactGroupQuickMoveRaw(ServoPositions() As Integer)
Parameters:	ServoPositions: Position (0~16383) to which servos will be moved. Expects a 16 element array of integers.
Return Value:	None

Command:	CompactGroupQuickMoveScaled(ServoPositions() As Integer)
Parameters:	ServoPositions: Scaled position (0~16383) to which servos will be moved. Expects a 16 element array of integers.
Return Value:	None

Command:	CompactGroupQuickMovePercent(ServoPositions() As Double)
Parameters:	ServoPositions: Scaled position (0~100) to which servos will be moved. Expects a 16 element array of doubles.
Return Value:	None

Command:	CompactGroupMoveRaw(ServoPositions() As Integer, ServoSpeed As Double)
Parameters:	ServoPositions: Position (0~16383) to which servos will be moved. Expects a 16 element array of doubles. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	CompactGroupMoveScaled(ServoPositions() As Integer, ServoSpeed As Double)
Parameters:	ServoPositions: Scaled position (0~16383) to which servos will be moved. Expects a 16 element array of doubles. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	CompactGroupMovePercent(ServoPositions() As Double, ServoSpeed As Double)
Parameters:	ServoPositions: Scaled position (0~100) to which servos will be moved. Expects a 16 element array of doubles. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	None

Command:	CompactGroupTimedMoveRaw(ServoPositions() As Integer, ServoTime As Double)
Parameters:	ServoPositions: Position (0~16383) to which servos will be moved. Expects a 16 element array of doubles. ServoTime: Length of time (0~23.9) in seconds for servo to move.
Return Value:	None

Command:	CompactGroupTimedMoveScaled(ServoPositions() As Integer, ServoTime As Double)
Parameters:	ServoPositions: Position (0~16383) to which servos will be moved. Expects a 16 element array of doubles. ServoTime: Length of time (0~23.9) in seconds for servo to move.
Return Value:	None

Command:	CompactGroupTimedMovePercent(ServoPositions() As Double, ServoTime As Double)
Parameters:	ServoPositions: Position (0~100) to which servos will be moved. Expects a 16 element array of doubles. ServoTime: Length of time (0~23.9) in seconds for servo to move.
Return Value:	None

Set Servo Settings Commands

Command:	ServoEnable(ServoNumber As Integer)
Parameters:	ServoNumber: ID (0~15) of servo to be enabled.
Return Value:	None

Command:	ServoDisable(ServoNumber As Integer)
Parameters:	ServoNumber: ID (0~15) of servo to be disabled.
Return Value:	None

Command:	ServoInvert(ServoNumber As Integer)
Parameters:	ServoNumber: ID (0~15) of servo to be inverted.
Return Value:	None

Command:	ServoUninvert(ServoNumber As Integer)
Parameters:	ServoNumber: ID (0~15) of servo to be normalized.
Return Value:	None

Command:	SetServoDisabledStateLow(ServoNumber As Integer)
Parameters:	ServoNumber: ID (0~15) of servo to set disabled state low.
Return Value:	None

Command:	SetServoDisabledStateHigh(ServoNumber As Integer)
Parameters:	ServoNumber: ID (0~15) of servo to set disabled state high.
Return Value:	None

Command:	SetMin(ServoNumber As Integer, ServoPosition As Integer)
Parameters:	ServoNumber: ID (0~15) of servo for which Min is being set. ServoPosition: Minimum value to be set.
Return Value:	None

Command:	SetMax(ServoNumber As Integer, ServoPosition As Integer)
Parameters:	ServoNumber: ID (0~15) of servo for which Max is being set. ServoPosition: Maximum value to be set.
Return Value:	None

Command:	SetStart(ServoNumber As Integer, ServoPosition As Integer)
Parameters:	ServoNumber: ID (0~15) of servo for which Start is being set. ServoPosition: Start value to be set.
Return Value:	None

Command:	SetSmoothingFactor(ServoNumber As Integer, SmoothingFactor As Integer)
Parameters:	ServoNumber: ID (0~15) of servo for which Start is being set. SmoothingFactor: (0~127) Sets Smoothing Factor for the specified servo.
Return Value:	None

Command:	SetMaxSpeed(ServoNumber As Integer, ServoSpeed As Integer)
Parameters:	ServoNumber: ID (0~15) of servo for which Speed is being set. ServoSpeed: Maximum speed of servo (1~200) in centi-seconds/60 degrees
Return Value:	None

Command:	SetMinCurrent(ServoNumber As Integer)
Parameters:	ServoNumber: ID (0~15) of servo for which Min is being set.
Return Value:	None

Command:	SetMaxCurrent(ServoNumber As Integer)
Parameters:	ServoNumber: ID (0~15) of servo for which Max is being set.
Return Value:	None

Command:	SetStartCurrent(ServoNumber As Integer)
Parameters:	ServoNumber: ID (0~15) of servo for which Start is being set.
Return Value:	None

Command:	SetPulseWidthMin(PulseWidthValue As Integer)
Parameters:	PulseWidthValue: Minimum (1~239) width of servo control pulses.
Return Value:	None

Command:	SetPulseWidthMax(PulseWidthValue As Integer)
Parameters:	PulseWidthValue: Maximum (1~239) width of servo control pulses.
Return Value:	None

Get Servo Settings Commands

Command:	GetServoEnableStatus(ServoNumber As Integer) As Integer
Parameters:	ServoNumber: ID (0~15) of servo for which enable status is being queried.
Return Value:	-1=Fail, 0=servo is disabled, 1=servo is enabled.

Command:	GetServoInvertStatus(ServoNumber As Integer) As Integer
Parameters:	ServoNumber: ID (0~15) of servo for which invert status is being queried.
Return Value:	-1=Fail, 0=servo is non-inverted, 1=servo is inverted.

Command:	GetServoDisabledState(ServoNumber As Integer) As Integer
Parameters:	ServoNumber: ID (0~15) of servo for which disable state is being queried.
Return Value:	-1=Fail, 0=servo disable state is low, 1=servo disable state is high.

Command:	GetCurrentPositionRaw(ServoNumber As Integer) As Integer
Parameters:	ServoNumber: ID (0~15) of servo for which position is being queried.
Return Value:	-1=Fail, Current position of servo (0~16383)

Command:	GetCurrentPositionScaled(ServoNumber As Integer) As Integer
Parameters:	ServoNumber: ID (0~15) of servo for which position is being queried.
Return Value:	-1=Fail, Current position of servo (0~16383)

Command:	GetCurrentPositionPercent(ServoNumber As Integer) As Double
Parameters:	ServoNumber: ID (0~15) of servo for which position is being queried.
Return Value:	-1=Fail, Current position of servo (0~100)

Command:	GetMinPosition(ServoNumber As Integer) As Integer
Parameters:	ServoNumber: ID (0~15) of servo for which Min is being queried.
Return Value:	-1=Fail, Minimum position of servo (0~16383)

Command:	GetMaxPosition(ServoNumber As Integer) As Integer
Parameters:	ServoNumber: ID (0~15) of servo for which Max is being queried.
Return Value:	-1=Fail, Maximum position of servo (0~16383)

Command:	GetStartPosition(ServoNumber As Integer) As Integer
Parameters:	ServoNumber: ID (0~15) of servo for which start position is being queried.
Return Value:	-1=Fail, Start position of servo (0~16383)

Command:	GetSmoothingFactor(ServoNumber As Integer) As Integer
Parameters:	ServoNumber: ID (0~15) of servo for which smoothing factor is being queried.
Return Value:	-1=Fail, Smoothing factor (0~127) of the particular servo.

Command:	GetMaxSpeed(ServoNumber As Integer) As Integer
Parameters:	ServoNumber: ID (0~15) of servo for which max speed is being queried.
Return Value:	-1=Fail, Maximum speed of servo (1~200) in centi-seconds/60 degrees

Command:	GetPulseWidthMin() As Integer
Parameters:	None
Return Value:	-1=Fail, Minimum (1~239) width of servo control pulses.

Command:	GetPulseWidthMax() As Integer
Parameters:	None
Return Value:	-1=Fail, Maximum (1~239) width of servo control pulses.

Input/Output Commands

Command:	ReadAD8(ADNum As Integer) As Integer
Parameters:	ADNum: (0~7) analog to digital input pin to read.
Return Value:	-1=Fail, Voltage (0~255) that is present on analog to digital input pin.

Command:	ReadAD10(ADNum As Integer) As Integer
Parameters:	ADNum: (0~7) analog to digital input pin to read.
Return Value:	-1=Fail, Voltage (0~1023) that is present on analog to digital input pin.

Command:	ReadDIOPinState(DIONum As Integer) As Integer
Parameters:	DIONum: (0~15) Digital I/O pin to read.
Return Value:	-1=Fail, 0=Low, 1=High

Command:	ReadDIOPinDirection(DIONum As Integer) As Integer
Parameters:	DIONum: (0~15) Digital I/O pin direction to read.
Return Value:	-1=Fail, 0=Pin is configured as an input, 1=Pin is configured as an output

Command:	ReadDIOPinChangeFlag(int DIONum) As Integer
Parameters:	DIONum: (0~15) Digital I/O pin change flag to read.
Return Value:	-1=Fail, 0=Pin has not changed, 1=Pin has changed

Command:	ReadDIOPinStartStateDirection(DIONum As Integer) As Integer
Parameters:	DIONum: (0~15) I/O pin to read starting state and direction.
Return Value:	-1=Fail, 0=Input no internal pull-up, 1=Input with internal pull-up, 2=Output low, 3=Output high

Command:	SetDIOPinLow(DIONum As Integer) As Integer
Parameters:	DIONum: (0~15) I/O pin to set state low.
Return Value:	0=Fail, Non-0=Success

Command:	SetDIOPinHigh(DIONum As Integer) As Integer
Parameters:	DIONum: (0~15) I/O pin to set state high.
Return Value:	0=Fail, Non-0=Success

Command:	SetDIOPinAsInput(DIONum As Integer) As Integer
Parameters:	DIONum: (0~15) I/O pin to set as input.
Return Value:	0=Fail, Non-0=Success

Command:	SetDIOPinAsOutput(DIONum As Integer) As Integer
Parameters:	DIONum: (0~15) I/O pin to set as output.
Return Value:	0=Fail, Non-0=Success

Command:	SetDIOPinStartStateDirection(DIONum As Integer, DIOState As Integer) As Integer
Parameters:	DIONum: (0~15) I/O pin to set start state and direction. DIOState: 0=input no internal pull-up, 1=input with internal pull-up, 2=output low, 3=output high.
Return Value:	0=Fail, Non-0=Success

Command:	SetAllDIOPinStartStatesDirections() As Integer
Parameters:	None
Return Value:	0=Fail, Non-0=Success

Servo Group Mask Commands

Command:	GroupMaskQuickMoveRaw(ServoPositions() As Integer)
Parameters:	ServoPositions: Position (0~16383) to which servos will be moved. Expects a 16 element array of integers. Values out of range are ignored and servo positions are not updated.
Return Value:	None

Command:	GroupMaskQuickMoveScaled(ServoPositions() As Integer)
Parameters:	ServoPositions: Scaled position (0~16383) to which servos will be moved. Expects a 16 element array of integers. Values out of range are ignored and servo positions are not updated.
Return Value:	None

Command:	GroupMaskQuickMovePercent(ServoPositions() As Double)
Parameters:	ServoPositions: Scaled position (0~100) to which servos will be moved. Expects a 16 element array of doubles. Values out of range are ignored and servo positions are not updated.
Return Value:	None

Command:	GroupMaskMoveRaw(ServoPositions() As Integer, ServoSpeed As Double)
Parameters:	ServoPositions: Position (0~16383) to which servos will be moved. Expects a 16 element array of doubles. Values out of range are ignored and servo positions are not updated. ServoSpeed: Speed (0~100) at which servos will move.
Return Value:	None

Command:	GroupMaskMoveScaled(ServoPositions() As Integer, ServoSpeed As Double)
Parameters:	ServoPositions: Scaled position (0~16383) to which servos will be moved. Expects a 16 element array of doubles. Values out of range are ignored and servo positions are not updated. ServoSpeed: Speed (0~100) at which servos will move.
Return Value:	None

Command:	GroupMaskMovePercent(ServoPositions() As Double, ServoSpeed As Double)
Parameters:	ServoPositions: Scaled position (0~100) to which servos will be moved. Expects a pointer to a 16 element array of doubles. Values out of range are ignored and servo positions are not updated. ServoSpeed: Speed (0~100) at which servos will move.
Return Value:	None

Command:	GroupMaskTimedMoveRaw(ServoPositions() As Integer, ServoTime As Double)
Parameters:	ServoPositions: Position (0~16383) to which servos will be moved. Expects a 16 element array of integers. Values out of range are ignored and servo positions are not updated. ServoTime: Length of time (0~23.9) in seconds for servos to move.
Return Value:	None

Command:	GroupMaskTimedMoveScaled(ServoPositions() As Integer, ServoTime As Double)
Parameters:	ServoPositions: Scaled position (0~16383) to which servos will be moved. Expects a 16 element array of integers. Values out of range are ignored and servo positions are not updated. ServoTime: Length of time (0~23.9) in seconds for servos to move.
Return Value:	None

Command:	GroupMaskTimedMovePercent(ServoPositions() As Double, ServoTime As Double)
Parameters:	ServoPositions: Scaled position (0~100) to which servos will be moved. Expects a pointer to a 16 element array of doubles. Values out of range are ignored and servo positions are not updated. ServoTime: Length of time (0~23.9) in seconds for servos to move.
Return Value:	None

Preset Commands

Command:	SetPresetServoData(PresetSlot As Integer, ServoPositions() As Double, EncodingFlags As Long, SkipFlags As Long)
Parameters:	PresetSlot: (0~63) Selects which preset slot to set preset servo data. ServoPos: Scaled position of servos, (0~16383) if binary scaled encoded, (0~100) if percentage scaled encoded. Expects a 16 element array of doubles. EncodingFlags: Bitmask used to determine encoding of servo data. Servo encoding is as follows: 0=servo is binary scaled encoding. 1=servo is percentage scaled encoding. The most significant bit (b15) selects servo S15 while the least significant bit (b0) selects S0. All other bits represent their respective servo. SkipFlags: Bitmask used to set the servo skip status. Skip encoding is as follows: 0=servo is not skipped. 1=servo is skipped. The most significant bit b15 selects servo S15 while the least significant bit (b0) selects S0. All other bits represent their respective servo.
Return Value:	None

Command:	GetPresetServoData(PresetSlot As Integer, ServoPositions() As Double, EncodingFlags As Long, SkipFlags As Long) As Boolean
Parameters:	PresetSlot: (0~63) Selects which preset slot to read preset servo data. ServoPositions: Expects a 16 element array of doubles. The array is filled with the preset data after the function returns. Detailed information about the preset data can be found in the ServoCenter 4.1 Protocol Manual under the “Get Preset Servo Data” command. EncodingFlags: Long variable to be filled with the encoding flags. Servo encoding is as follows: 0=servo is binary scaled encoding. 1=servo is percentage scaled encoding. The most significant bit b15 selects servo S15 while the least significant bit (b0) selects S0. All other bits represent their respective servo. SkipFlags: Long variable to be filled with the skip flags. Skip encoding is as follows: 0=servo is not skipped. 1=servo is skipped. The most significant bit b15 selects servo S15 while the least significant bit (b0) selects S0. All other bits represent their respective servo.
Return Value:	False=Fail, True=Success

Command:	SetPresetControlData(PresetSlot As Integer, PresetData As ControlData)
Parameters:	PresetSlot: (0~63) Selects which preset slot to set preset control data. PresetData: Expects a custom type consisting of four longs; ServoEnabledFlags, DIOSkipFlags, DIODirections, and DIOValues. The custom type is accessible within the control. ServoEnabledFlags: 16-bit value with each bit corresponding to the enabled state of a servo. Thus, b0 is servo S0's enabled state, b1 is servo S1's enabled state, etc. DIOSkipFlags: 16-bit value with each bit corresponding to the skip state of a digital I/O channel. Thus, b0 is DIO0's skip state, b1 is DIO1's skip state, etc. When the skip state bit is high for a channel, the state and direction for that digital I/O channel is unmodified. This allows digital I/O presets to be effectively masked and layered. DIODirections: 16-bit value with each bit corresponding to the pin direction of a digital I/O channel. Thus, b0 is DIO0's direction value, b1 is DIO1's direction value, etc. When the direction bit is 0, the direction for that digital I/O channel is set as an input. When the direction bit is 1, the direction for that digital I/O channel is set as an output. DIOValues: 16-bit value with each bit corresponding to the pin state of a digital I/O channel. Thus, b0 is DIO0's state value, b1 is DIO1's state value, etc. When the state bit is 0, the state for that digital I/O channel is set low. When the state bit is 1, the state for that digital I/O channel is set high. When a pin is configured as an input, the value bit controls the application of an internal pull-up resistance for each channel.
Return Value:	None

Command:	GetPresetControlData(PresetSlot As Integer, PresetData As ControlData) As Integer
Parameters:	PresetSlot: (0~63) Selects which preset slot to read preset control data. PresetData: Expects a ControlData type variable that will be filled in with the preset control data once the function returns.
Return Value:	False=Fail, True=Success

Command:	SetPresetName(PresetSlot As Integer, PresetName As String)
Parameters:	PresetSlot: (0~63) Selects which preset slot is named. PresetName: Expects a string up to 16 characters long.
Return Value:	None

Command:	GetPresetName(PresetSlot As Integer, PresetName As String) As Boolean
Parameters:	PresetSlot: (0~63) Selects which preset slot name is read. PresetName: The string will contain the name of the selected preset if the function returns successfully.
Return Value:	False=Fail, True=Success

Command:	QuickLoadPreset(PresetSlot As Integer)
Parameters:	PresetSlot: (0~63) Specifies the preset to be loaded.
Return Value:	None

Command:	CrossfadePreset(PresetSlot As Integer, CrossfadeTime As Double)
Parameters:	PresetSlot: (0~63) Specifies the preset to be loaded. CrossfadeTime: (0~23.9) Time in seconds the preset is to be cross-faded from the current servo positions.
Return Value:	None

Command:	StoreCurrentAsPreset(PresetSlot As Integer)
Parameters:	PresetSlot: (0~63) Specifies which preset slot to save current settings.
Return Value:	None

Command:	InitializePreset(PresetSlot As Integer)
Parameters:	PresetSlot: (0~63) Specifies which preset slot to initialize.
Return Value:	None

Command:	ReadPresetEEPROM(Page As Integer, EEPROMData() As Byte) As Boolean
Parameters:	Page: (0~127) Specifies which preset EEPROM page to read. EEPROMData: A 32 element byte array. The array is filled with the specified page if the function returns successfully.
Return Value:	False=Fail, True=Success

Command:	WritePresetEEPROM(Page As Integer, EEPROMData() As Byte) As Boolean
Parameters:	Page: (0~127) Specifies which preset EEPROM page to write. EEPROMData: A 32 element byte array filled with the contents to be written to the specified preset EEPROM page.
Return Value:	False=Fail, True=Success

Sequencer / BASIC Interpreter Commands

Command:	ReadSequencerEEPROM(Page As Integer, EEPROMData() As Byte) As Boolean
Parameters:	Page: (0~127) Specifies which sequencer EEPROM page to read. EEPROMData: A 32 element byte array. The array is filled with the specified page if the function returns successfully.
Return Value:	False=Fail, True=Success

Command:	WriteSequencerEEPROM(Page As Integer, EEPROMData() As Byte) As Boolean
Parameters:	Page: (0~127) Specifies which sequencer EEPROM page to write. EEPROMData: A 32 element byte array filled with the contents to be written to the specified sequencer EEPROM page.
Return Value:	False=Fail, True=Success

Command:	StartSequencer(SequenceArg As Integer)
Parameters:	SequenceArg: (0~255) Single byte parameter argument passed to the sequencer.
Return Value:	None

Command:	StopSequencer()
Parameters:	None
Return Value:	None

Command:	ResetSequencer()
Parameters:	None
Return Value:	None

Command:	WriteCharToSequencer(SequencerChar As Integer)
Parameters:	SequencerChar: (0~255) Single byte parameter to be passed to the sequencer's character input buffer.
Return Value:	None

Command:	SetSequencerStartupMode(SequencerStartupMode As Integer)
Parameters:	SequencerStartupMode: (0~1) 0=The stored sequencer program can only be started by the reception of the Start Sequencer command. 1=The stored sequencer program is automatically started when the board is reset / powered up or when the Start Sequencer command is received. This effectively allows the SC4.1 to be used in stand-alone applications.
Return Value:	None

Command:	GetSequencerStartupMode() As Integer
Parameters:	None
Return Value:	-1=Fail. 0=The stored sequencer program can only be started by the reception of the Start Sequencer command. 1=The stored sequencer program is automatically started when the board is reset / powered up or when the Start Sequencer command is received.

Command:	GetSequencerStatus() As Integer
Parameters:	None
Return Value:	-1=Fail. 0=Sequencer Idle / Stopped. 1=Sequencer Running. 2 =Sequencer Error / Stopped

Command:	GetSequencerLastError() As Integer
Parameters:	None
Return Value:	-1=Fail. Non-negative=The sequencer error byte value.

General Commands

Command:	SetLEDMode(LEDMode As Integer)
Parameters:	LEDMode: (0~7) specifies the LED display mode for LED1 and LED2.
Return Value:	None

Command:	SetWatchdogTime(WdTimeTenths As Integer)
Parameters:	WdTimeTenths: (0~239) Sets the internal watchdog time to the value specified by WdTimeTenths as measured in 1/10th second increments.
Return Value:	None

Command:	CommitSettings()
Parameters:	None
Return Value:	None

Command:	LoadFactorySettings()
Parameters:	None
Return Value:	None

Command:	ResetAsStartup()
Parameters:	None
Return Value:	None

Command:	DisplayVersion(Version As String)
Parameters:	Version: Will be filled with the ServoCenter's version information once the function returns.
Return Value:	False=Fail, True=Success

2.5 Programming in Visual Basic 6.0 with the YEIServoControl

The following code examples illustrate the use of the YEIServoControl in Visual Basic 6.0.

Here is an example of how to initialize your ServoCenter 4.1 Board:

```
Private Sub Form_Load()
    YEIServoCtrl1.BaudRate = vb9600    ' set the baud to 9600 bps
    YEIServoCtrl1.BoardID = vbBoard0   ' set board id to 0
    YEIServoCtrl1.InitBoard     'initialize the control
End Sub
```

Here is an example of how to move a servo by clicking on a button:

```
Private Sub cmdMoveServo_Click()
Call YEIServoCtrl1.QuickMoveRaw(0, 100)
'Move servo 0 to raw position 100
End Sub
```

Here is an example of how to show the ServoCenter control panel:

```
Private Sub cmdShowConfig_Click()
    YEIServoCtrl1.ShowControlPanel    'show control panel
End Sub
```

The DirectSerial.vbp project, located on the ServoCenter 4.1 CD, is a sample Visual Basic project that demonstrates usage of the YEIServoControl in the Visual Basic 6.0 environment.

3. Programming With the ServoCenter 4.1 DLL

ServoCenter 4.1 comes packaged with the yeisrvo.dll runtime library, which gives programmers access to low-level predefined functions that can be used with the ServoCenter 4.1 controller board. This section covers the capabilities of the DLL, installing the DLL, and writing programs using the DLL functions.

3.1 ServoCenter 4.1 DLL Functional Overview

The functions provided by the ServoCenter 4.1 DLL correspond with the ServoCenter 4.1 controller board commands detailed in Section 4.1.4, except as noted in the table descriptions listed below.

Port Functions

Function:	void InitPort(int Comm, long BaudRate)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BaudRate: Data rate at which the serial port will communicate.
Return Value:	0 – Success Other - Error

Normal Movement Servo Commands

Function:	int QuickMoveRaw(int Comm, int BoardNum, int ServoNum, int ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo whose position is to be changed. ServoPos: Raw position (0~16383) to which servo will be moved.
Return Value:	0=Fail, Non-0=Success

Function:	int QuickMoveScaled(int Comm, int BoardNum, int ServoNum, int ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo whose position is to be changed. ServoPos: Scaled position (0~16383) to which servo will be moved.
Return Value:	0=Fail, Non-0=Success

Function:	int QuickMovePercent(int Comm, int BoardNum, int ServoNum, float ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo whose position is to be changed. ServoPos: Scaled position (0~100) to which servo will be moved.
Return Value:	0=Fail, Non-0=Success

Function:	int MoveRaw(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int MoveRawCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Units (0~16383) servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int MoveRawCCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Units (0~16383) servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int MoveScaled(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int MoveScaledCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int MoveScaledCCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int MovePercent(int Comm, int BoardNum, int ServoNum, float ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~100) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int MovePercentCW(int Comm, int BoardNum, int ServoNum, float ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~100) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int MovePercentCCW(int Comm, int BoardNum, int ServoNum, float ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~100) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int TimedMoveRaw(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Units (0~16383) servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int TimedMoveRawCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Units (0~16383) servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int TimedMoveRawCCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Units (0~16383) servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int TimedMoveScaled(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~16383) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int TimedMoveScaledCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~16383) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int TimedMoveScaledCCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~16383) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int TimedMovePercent(int Comm, int BoardNum, int ServoNum, float ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~100) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int TimedMovePercentCW(int Comm, int BoardNum, int ServoNum, float ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~100) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int TimedMovePercentCCW(int Comm, int BoardNum, int ServoNum, float ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~100) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Normal Movement Group Commands

Function:	int GroupQuickMoveRaw(int Comm, int BoardNum, int * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupQuickMoveScaled(int Comm, int BoardNum, int * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Scaled position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupQuickMovePercent(int Comm, int BoardNum, float * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Scaled position (0~100) to which servos will be moved. Expects a pointer to a 16 element array of floats.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupMoveRaw(int Comm, int BoardNum, int * ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupMoveScaled(int Comm, int BoardNum, int * ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Scaled position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupMovePercent(int Comm, int BoardNum, float * ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Scaled position (0~100) to which servos will be moved. Expects a pointer to a 16 element array of floats. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupTimedMoveRaw(int Comm, int BoardNum, int * ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. ServoTime: Length of time (0~23.9) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupTimedMoveScaled(int Comm, int BoardNum, int * ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. ServoTime: Length of time (0~23.9) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupTimedMovePercent(int Comm, int BoardNum, float * ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Position (0~100) to which servos will be moved. Expects a pointer to a 16 element array of floats. ServoTime: Length of time (0~23.9) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Compact Movement Servo Commands

Function:	int CompactQuickMoveRaw(int Comm, int BoardNum, int ServoNum, int ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo whose position is to be changed. ServoPos: Raw position (0~16383) to which servo will be moved.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactQuickMoveScaled(int Comm, int BoardNum, int ServoNum, int ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo whose position is to be changed. ServoPos: Scaled position (0~16383) to which servo will be moved.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactQuickMovePercent(int Comm, int BoardNum, int ServoNum, float ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo whose position is to be changed. ServoPos: Scaled position (0~100) to which servo will be moved.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactMoveRaw(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactMoveRawCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Units (0~16383) servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactMoveRawCCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Units (0~16383) servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactMoveScaled(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactMoveScaledCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactMoveScaledCCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~16383) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactMovePercent(int Comm, int BoardNum, int ServoNum, float ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~100) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactMovePercentCW(int Comm, int BoardNum, int ServoNum, float ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~100) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactMovePercentCCW(int Comm, int BoardNum, int ServoNum, float ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~100) to which servo will be moved. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactTimedMoveRaw(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Units (0~16383) servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactTimedMoveRawCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Units (0~16383) servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactTimedMoveRawCCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Units (0~16383) servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactTimedMoveScaled(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~16383) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactTimedMoveScaledCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~16383) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactTimedMoveScaledCCW(int Comm, int BoardNum, int ServoNum, int ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~16383) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactTimedMovePercent(int Comm, int BoardNum, int ServoNum, float ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~100) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactTimedMovePercentCW(int Comm, int BoardNum, int ServoNum, float ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~100) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactTimedMovePercentCCW(int Comm, int BoardNum, int ServoNum, float ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be moved. ServoPos: Scaled position (0~100) to which servo will be moved. ServoTime: Length of time (0~239.99) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Compact Movement Group Commands

Function:	int CompactGroupQuickMoveRaw(int Comm, int BoardNum, int * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactGroupQuickMoveScaled(int Comm, int BoardNum, int * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Scaled position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactGroupQuickMovePercent(int Comm, int BoardNum, float * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Scaled position (0~100) to which servos will be moved. Expects a pointer to a 16 element array of floats.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactGroupMoveRaw(int Comm, int BoardNum, int * ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactGroupMoveScaled(int Comm, int BoardNum, int * ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Scaled position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactGroupMovePercent(int Comm, int BoardNum, float * ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Scaled position (0~100) to which servos will be moved. Expects a pointer to a 16 element array of floats. ServoSpeed: Speed (0~100) at which servo will move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactGroupTimedMoveRaw(int Comm, int BoardNum, int * ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. ServoTime: Length of time (0~23.9) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactGroupTimedMoveScaled(int Comm, int BoardNum, int * ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. ServoTime: Length of time (0~23.9) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Function:	int CompactGroupTimedMovePercent(int Comm, int BoardNum, float * ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Position (0~100) to which servos will be moved. Expects a pointer to a 16 element array of floats. ServoTime: Length of time (0~23.9) in seconds for servo to move.
Return Value:	0=Fail, Non-0=Success

Set Servo Settings Commands

Function:	int ServoEnable(int Comm, int BoardNum, int ServoNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be enabled.
Return Value:	0=Fail, Non-0=Success

Function:	int ServoDisable(int Comm, int BoardNum, int ServoNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be disabled.
Return Value:	0=Fail, Non-0=Success

Function:	int ServoInvert(int Comm, int BoardNum, int ServoNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be inverted.
Return Value:	0=Fail, Non-0=Success

Function:	int ServoUninvert(int Comm, int BoardNum, int ServoNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to be normalized.
Return Value:	0=Fail, Non-0=Success

Function:	int SetServoDisabledStateLow(int Comm, int BoardNum, int ServoNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to set disabled state low.
Return Value:	0=Fail, Non-0=Success

Function:	int SetServoDisabledStateHigh(int Comm, int BoardNum, int ServoNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo to set disabled state high.
Return Value:	0=Fail, Non-0=Success

Function:	int SetMin(int Comm, int BoardNum, int ServoNum, int ServoMinPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which Min is being set. ServoPos: Minimum value to be set.
Return Value:	0=Fail, Non-0=Success

Function:	int SetMax(int Comm, int BoardNum, int ServoNum, int ServoMaxPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which Max is being set. ServoPos: Maximum value to be set.
Return Value:	0=Fail, Non-0=Success

Function:	int SetStart(int Comm, int BoardNum, int ServoNum, int ServoStartPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which Start is being set. ServoPos: Start value to be set.
Return Value:	0=Fail, Non-0=Success

Function:	int SetSmoothingFactor(int Comm, int BoardNum, int ServoNum, int SmoothingFactor)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which Start is being set. SmoothingFactor: (0~127) Sets Smoothing Factor for the specified servo.
Return Value:	0=Fail, Non-0=Success

Function:	int SetMaxSpeed(int Comm, int BoardNum, int ServoNum, int ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which Speed is being set. ServoSpeed: Maximum speed of servo (1~200) in centi-seconds/60 degrees
Return Value:	0=Fail, Non-0=Success

Function:	int SetMinCurrent(int Comm, int BoardNum, int ServoNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which Min is being set.
Return Value:	0=Fail, Non-0=Success

Function:	int SetMaxCurrent(int Comm, int BoardNum, int ServoNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which Max is being set.
Return Value:	0=Fail, Non-0=Success

Function:	int SetStartCurrent(int Comm, int BoardNum, int ServoNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which Start is being set.
Return Value:	0=Fail, Non-0=Success

Function:	int SetPulseWidthMin(int Comm, int BoardNum, int WidthVal)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. WidthVal: Minimum (1~239) width of servo control pulses.
Return Value:	0=Fail, Non-0=Success

Function:	int SetPulseWidthMax(int Comm, int BoardNum, int WidthVal)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. WidthVal: Maximum (1~239) width of servo control pulses.
Return Value:	0=Fail, Non-0=Success

Get Servo Settings Commands

Function:	int GetServoEnableStatus(int Comm, int BoardNum, int ServoNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which enable status is being queried.
Return Value:	-1=Fail, 0=servo is disabled, 1=servo is enabled.

Function:	int GetServoInvertStatus(int Comm, int BoardNum, int ServoNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which invert status is being queried.
Return Value:	-1=Fail, 0=servo is non-inverted, 1=servo is inverted.

Function:	int GetServoDisabledState(int Comm, int BoardNum, int ServoNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which disable state is being queried.
Return Value:	-1=Fail, 0=servo disable state is low, 1=servo disable state is high.

Function:	int GetCurrentPositionRaw(int Comm, int BoardNum, int ServoNum, int * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which position is being queried. ServoPos: Pointer which will be filled with the servo position (0~16383) once the function returns.
Return Value:	0=Fail, Non-0=Success

Function:	int GetCurrentPositionScaled(int Comm, int BoardNum, int ServoNum, int * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which position is being queried. ServoPos: Pointer which will be filled with the servo position (0~16383) once the function returns.
Return Value:	0=Fail, Non-0=Success

Function:	float GetCurrentPositionPercent(int Comm, int BoardNum, int ServoNum, float * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which position is being queried. ServoPos: Pointer which will be filled with the servo position (0~100) once the function returns.
Return Value:	0=Fail, Non-0=Success

Function:	int GetMinPosition(int Comm, int BoardNum, int ServoNum, int * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which Min is being queried. ServoPos: Pointer which will be filled with the servo position (0~16383) once the function returns.
Return Value:	0=Fail, Non-0=Success

Function:	Int GetMaxPosition(int Comm, int BoardNum, int ServoNum, int * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which Max is being queried. ServoPos: Pointer which will be filled with the servo position (0~16383) once the function returns.
Return Value:	0=Fail, Non-0=Success

Function:	int GetStartPosition(int Comm, int BoardNum, int ServoNum, int * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which start position is being queried. ServoPos: Pointer which will be filled with the servo position (0~100) once the function returns.
Return Value:	0=Fail, Non-0=Success

Function:	int GetSmoothingFactor(int Comm, int BoardNum, int ServoNum, int * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which smoothing factor is being queried.
Return Value:	-1=Fail, Smoothing factor (0~127) of the particular servo.

Function:	int GetMaxSpeed(int Comm, int BoardNum, int ServoNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoNum: ID (0~15) of servo for which max speed is being queried.
Return Value:	-1=Fail, Maximum speed of servo (1~200) in centi-seconds/60 degrees

Function:	int GetPulseWidthMin(int Comm, int BoardNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent.
Return Value:	-1=Fail, Minimum (1~239) width of servo control pulses.

Function:	int GetPulseWidthMax(int Comm, int BoardNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent.
Return Value:	-1=Fail, Maximum (1~239) width of servo control pulses.

Input/Output Commands

Function:	int ReadAD8(int Comm, int BoardNum, int ADNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ADNum: (0~7) analog to digital input pin to read.
Return Value:	-1=Fail, Voltage (0~255) that is present on analog to digital input pin.

Function:	int ReadAD10(int Comm, int BoardNum, int ADNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ADNum: (0~7) analog to digital input pin to read.
Return Value:	-1=Fail, Voltage (0~1023) that is present on analog to digital input pin.

Function:	int ReadDIOPinState(int Comm, int BoardNum, int DIONum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. DIONum: (0~15) Digital I/O pin to read.
Return Value:	-1=Fail, 0=Low, 1=High

Function:	int ReadDIOPinDirection(int Comm, int BoardNum, int DIONum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. DIONum: (0~15) Digital I/O pin direction to read.
Return Value:	-1=Fail, 0=Pin is configured as an input, 1=Pin is configured as an output

Function:	int ReadDIOPinChangeFlag(int Comm, int BoardNum, int DIONum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. DIONum: (0~15) Digital I/O pin change flag to read.
Return Value:	-1=Fail, 0=Pin has not changed, 1=Pin has changed

Function:	int ReadDIOPinStartStateDirection(int Comm, int BoardNum, int DIONum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. DIONum: (0~15) I/O pin to read starting state and direction.
Return Value:	-1=Fail, 0=Input no internal pull-up, 1=Input with internal pull-up, 2=Output low, 3=Output high

Function:	int SetDIOPinLow(int Comm, int BoardNum, int DIONum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. DIONum: (0~15) I/O pin to set state low.
Return Value:	0=Fail, Non-0=Success

Function:	int SetDIOPinHigh(int Comm, int BoardNum, int DIONum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. DIONum: (0~15) I/O pin to set state high.
Return Value:	0=Fail, Non-0=Success

Function:	int SetDIOPinAsInput(int Comm, int BoardNum, int DIONum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. DIONum: (0~15) I/O pin to set as input.
Return Value:	0=Fail, Non-0=Success

Function:	int SetDIOPinAsOutput(int Comm, int BoardNum, int DIONum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. DIONum: (0~15) I/O pin to set as output.
Return Value:	0=Fail, Non-0=Success

Function:	int SetDIOPinStartStateDirection(int Comm, int BoardNum, int DIONum, int DIOState)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. DIONum: (0~15) I/O pin to set start state and direction. DIOState: 0=input no internal pull-up, 1=input with internal pull-up, 2=output low, 3=output high.
Return Value:	0=Fail, Non-0=Success

Function:	int SetAllDIOPinStartStatesDirections(int Comm, int BoardNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent.
Return Value:	0=Fail, Non-0=Success

Servo Group Mask Commands

Function:	int GroupMaskQuickMoveRaw(int Comm, int BoardNum, int * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. Values out of range are ignored and servo positions are not updated.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupMaskQuickMoveScaled(int Comm, int BoardNum, int * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Scaled position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. Values out of range are ignored and servo positions are not updated.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupMaskQuickMovePercent(int Comm, int BoardNum, float * ServoPos)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Scaled position (0~100) to which servos will be moved. Expects a pointer to a 16 element array of integers. Values out of range are ignored and servo positions are not updated.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupMaskMoveRaw(int Comm, int BoardNum, int * ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. Values out of range are ignored and servo positions are not updated. ServoSpeed: Speed (0~100) at which servos will move.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupMaskMoveScaled(int Comm, int BoardNum, int * ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Scaled position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. Values out of range are ignored and servo positions are not updated. ServoSpeed: Speed (0~100) at which servos will move.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupMaskMovePercent(int Comm, int BoardNum, float * ServoPos, float ServoSpeed)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Scaled position (0~100) to which servos will be moved. Expects a pointer to a 16 element array of floats. Values out of range are ignored and servo positions are not updated. ServoSpeed: Speed (0~100) at which servos will move.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupMaskTimedMoveRaw(int Comm, int BoardNum, int * ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. Values out of range are ignored and servo positions are not updated. ServoTime: Length of time (0~23.9) in seconds for servos to move.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupMaskTimedMoveScaled(int Comm, int BoardNum, int * ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Scaled position (0~16383) to which servos will be moved. Expects a pointer to a 16 element array of integers. Values out of range are ignored and servo positions are not updated. ServoTime: Length of time (0~23.9) in seconds for servos to move.
Return Value:	0=Fail, Non-0=Success

Function:	int GroupMaskTimedMovePercent(int Comm, int BoardNum, float * ServoPos, float ServoTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. ServoPos: Scaled position (0~100) to which servos will be moved. Expects a pointer to a 16 element array of floats. Values out of range are ignored and servo positions are not updated. ServoTime: Length of time (0~23.9) in seconds for servos to move.
Return Value:	0=Fail, Non-0=Success

Preset Commands

Function:	int SetPresetServoData(int Comm, int BoardNum, int PresetSlot, float * ServoPos, int EncodingMask, int SkipFlags)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. PresetSlot: (0~63) Selects which preset slot to set preset servo data. ServoPos: Scaled position of servos, (0~16383) if binary scaled encoded, (0~100) if percentage scaled encoded. Expects a pointer to a 16 element array of floats. Values out of range are ignored and servo positions are not updated. This allows servo presets to be effectively masked and layered. EncodingMask: Bitmask used to determine encoding of servo data. Servo encoding is as follows: 0=servo is binary scaled encoding. 1=servo is percentage scaled encoding. The most significant bit of EncodingMask(b15) selects servo S15 while the least significant bit (b0) selects S0. All other bits represent their respective servo. SkipFlags: Bitmask used to determine skip flags of the specified preset . Skip flag encoding is as follows: 0=servo is not skipped. 1=servo is skipped. The most significant bit (b15) represents servo S15 while the least significant bit (b0) represents S0. All other bits represent their respective servo.
Return Value:	0=Fail, Non-0=Success

Function:	int GetPresetServoData(int Comm, int BoardNum, int PresetSlot, float * ServoPos, int * EncodingMask, int * SkipFlags)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. PresetSlot: (0~63) Selects which preset slot to read preset servo data. ServoPos: Expects a pointer to a 16 element array of floats. The array is filled with the preset data after the function returns. Detailed information about the preset data can be found in the ServoCenter 4.1 Protocol Manual under the “Set Preset Servo Data” command. EncodingMask: Expects a pointer to an integer to be filled with the preset servo position encoding mask. Servo encoding is as follows: 0=servo is binary scaled encoding. 1=servo is percentage scaled encoding. The most significant bit of EncodingMask(b15) selects servo S15 while the least significant bit (b0) selects S0. All other bits represent their respective servo. SkipFlags: Expects a pointer to an integer to be filled with the skip flags of the specified preset. Skip flag encoding is as follows: 0=servo is not skipped. 1=servo is skipped. The most significant bit (b15) represents servo S15 while the least significant bit (b0) represents S0. All other bits represent their respective servo.
Return Value:	0=Fail, Non-0=Success

Function:	int SetPresetControlData(int Comm, int BoardNum, int PresetSlot, ControlData PresetData)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. PresetSlot: (0~63) Selects which preset slot to set preset control data. PresetData: Expects a structure consisting of four integers; ServoEnabledFlags, DIOSkipFlags, DIODirections, and DIOValues. ServoEnabledFlags: 16-bit value with each bit corresponding to the enabled state of a servo. Thus, b0 is servo S0's enabled state, b1 is servo S1's enabled state, etc. DIOSkipFlags: 16-bit value with each bit corresponding to the skip state of a digital I/O channel. Thus, b0 is DIO0's skip state, b1 is DIO1's skip state, etc. When the skip state bit is high for a channel, the state and direction for that digital I/O channel is unmodified. This allows digital I/O presets to be effectively masked and layered. DIODirections: 16-bit value with each bit corresponding to the pin direction of a digital I/O channel. Thus, b0 is DIO0's direction value, b1 is DIO1's direction value, etc. When the direction bit is 0, the direction for that digital I/O channel is set as an input. When the direction bit is 1, the direction for that digital I/O channel is set as an output. DIOValues: 16-bit value with each bit corresponding to the pin state of a digital I/O channel. Thus, b0 is DIO0's state value, b1 is DIO1's state value, etc. When the state bit is 0, the state for that digital I/O channel is set low. When the state bit is 1, the state for that digital I/O channel is set high. When a pin is configured as an input, the value bit controls the application of an internal pull-up resistance for each channel.
Return Value:	0=Fail, Non-0=Success

Function:	int GetPresetControlData(int Comm, int BoardNum, int PresetSlot, ControlData * PresetData)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. PresetSlot: (0~63) Selects which preset slot to read preset control data. PresetData: Expects a pointer to a structure consisting of four integers; ServoEnabledFlags, DIOSkipFlags, DIODirections, and DIOValues. The struture is filled with the preset control data once the function has returned. ServoEnabledFlags: 16-bit value with each bit corresponding to the enabled state of a servo. Thus, b0 is servo S0's enabled state, b1 is servo S1's enabled state, etc. DIOSkipFlags: 16-bit value with each bit corresponding to the skip state of a digital I/O channel. Thus, b0 is DIO0's skip state, b1 is DIO1's skip state, etc. When the skip state bit is high for a channel, the state and direction for that digital I/O channel is unmodified. This allows digital I/O presets to be effectively masked and layered. DIODirections: 16-bit value with each bit corresponding to the pin direction of a digital I/O channel. Thus, b0 is DIO0's direction value, b1 is DIO1's direction value, etc. When the direction bit is 0, the direction for that digital I/O channel is set as an input. When the direction bit is 1, the direction for that digital I/O channel is set as an output. DIOValues: 16-bit value with each bit corresponding to the pin state of a digital I/O channel. Thus, b0 is DIO0's state value, b1 is DIO1's state value, etc. When the state bit is 0, the state for that digital I/O channel is set low. When the state bit is 1, the state for that digital I/O channel is set high. When a pin is configured as an input, the value bit controls the application of an internal pull-up resistance for each channel.
Return Value:	0=Fail, Non-0=Success

Function:	int SetPresetName(int Comm, int BoardNum, int PresetSlot, char * PresetName)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. PresetSlot: (0~63) Selects which preset slot is named. PresetName: Expects a pointer to a null terminated string up to 16 characters long.
Return Value:	0=Fail, Non-0=Success

Function:	int GetPresetName(int Comm, int BoardNum, int PresetSlot, char * PresetName)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. PresetSlot: (0~63) Selects which preset slot is named. PresetName: Expects a pointer to an array of characters 17 elements long. The array is filled with the null terminated string of the selected preset if the function returns successfully.
Return Value:	0=Fail, Non-0=Success

Function:	int QuickLoadPreset(int Comm, int BoardNum, int PresetSlot)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. PresetSlot: (0~63) Specifies the preset to be loaded.
Return Value:	0=Fail, Non-0=Success

Function:	int CrossfadePreset(int Comm, int BoardNum, int PresetSlot, float CrossfadeTime)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. PresetSlot: (0~63) Specifies the preset to be loaded. CrossfadeTime: (0~23.9) Time in seconds the preset is to be cross-faded from the current servo positions.
Return Value:	0=Fail, Non-0=Success

Function:	int StoreCurrentAsPreset(int Comm, int BoardNum, int PresetSlot)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. PresetSlot: (0~63) Specifies which preset slot to save current settings.
Return Value:	0=Fail, Non-0=Success

Function:	int InitializePreset(int Comm, int BoardNum, int PresetSlot)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. PresetSlot: (0~63) Specifies which preset slot to initialize.
Return Value:	0=Fail, Non-0=Success

Function:	int ReadPresetEEPROM(int Comm, int BoardNum, int Page, unsigned char * EEPROMData)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. Page: (0~127) Specifies which preset EEPROM page to read. EEPROMData: A pointer to a 32 element unsigned char array. The array is filled with the specified page if the function returns successfully.
Return Value:	0=Fail, Non-0=Success

Function:	int WritePresetEEPROM(int Comm, int BoardNum, int Page, unsigned char * EEPROMData)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. Page: (0~127) Specifies which preset EEPROM page to write. EEPROMData: A pointer to a 32 element unsigned char array filled with the contents to be written to the specified preset EEPROM page.
Return Value:	0=Fail, Non-0=Success

Sequencer / BASIC Interpreter Commands

Function:	int ReadSequencerEEPROM(int Comm, int BoardNum, int Page, unsigned char * EEPROMData)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. Page: (0~127) Specifies which sequencer EEPROM page to read. EEPROMData: A pointer to a 32 element unsigned char array. The array is filled with the specified page if the function returns successfully.
Return Value:	0=Fail, Non-0=Success

Function:	int WriteSequencerEEPROM(int Comm, int BoardNum, int Page, unsigned char * EEPROMData)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. Page: (0~127) Specifies which sequencer EEPROM page to write. EEPROMData: A pointer to a 32 element unsigned char array filled with the contents to be written to the specified sequencer EEPROM page.
Return Value:	0=Fail, Non-0=Success

Function:	int StartSequencer(int Comm, int BoardNum, int SequenceArg)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. SequenceArg: (0~255) Single byte parameter argument passed to the sequencer.
Return Value:	0=Fail, Non-0=Success

Function:	int StopSequencer(int Comm, int BoardNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent.
Return Value:	0=Fail, Non-0=Success

Function:	int ResetSequencer(int Comm, int BoardNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent.
Return Value:	0=Fail, Non-0=Success

Function:	int WriteCharToSequencer(int Comm, int BoardNum, int DataByte)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. DataByte: (0~255) Single byte parameter to be passed to the sequencer's character input buffer.
Return Value:	0=Fail, Non-0=Success

Function:	int SetSequencerStartupMode(int Comm, int BoardNum, int SequencerStartupMode)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. SequencerStartupMode: (0~1) 0=The stored sequencer program can only be started by the reception of the Start Sequencer command. 1=The stored sequencer program is automatically started when the board is reset / powered up or when the Start Sequencer command is received. This effectively allows the SC4.1 to be used in stand-alone applications.
Return Value:	0=Fail, Non-0=Success

Function:	int GetSequencerStartupMode(int Comm, int BoardNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent.
Return Value:	-1=Fail. 0=The stored sequencer program can only be started by the reception of the Start Sequencer command. 1=The stored sequencer program is automatically started when the board is reset / powered up or when the Start Sequencer command is received.

Function:	int GetSequencerStatus(int Comm, int BoardNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent.
Return Value:	-1=Fail. 0=Sequencer Idle / Stopped. 1=Sequencer Running. 2 =Sequencer Error / Stopped

Function:	int GetSequencerLastError(int Comm, int BoardNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent.
Return Value:	-1=Fail. Non-negative=The sequencer error byte value.

General Commands

Function:	int SetLEDMode(int Comm, int BoardNum, int Mode)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. Mode: (0~7) specifies the LED display mode for LED1 and LED2.
Return Value:	0=Fail, Non-0=Success

Function:	int SetWatchdogTime(int Comm, int BoardNum, int WdTimeTenths)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. WdTimeTenths: (0~239) Sets the internal watchdog time to the value specified by WdTimeTenths as measured in 1/10th second increments.
Return Value:	0=Fail, Non-0=Success

Function:	void CommitSettings(int Comm, int BoardNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent.
Return Value:	0=Fail, Non-0=Success

Function:	void LoadFactorySettings(int Comm, int BoardNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent.
Return Value:	0=Fail, Non-0=Success

Function:	void ResetAsStartup(int Comm, int BoardNum)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent.
Return Value:	0=Fail, Non-0=Success

Function:	int DisplayVersion(int Comm, int BoardNum, char * VersionInfo)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. BoardNum: ID (0~15) of ServoCenter 4.1 Board to which the command will be sent. VersionInfo: 64 element char buffer to hold version information.
Return Value:	Length of VersionInfo string

Function:	void CloseCom(int Comm)
Parameters:	Comm: Communications Port to be closed.
Return Value:	None

Function:	void CloseAllComs()
Parameters:	None
Return Value:	None

Function:	void SetComTimeouts(int Comm, int ReadMultiplier, int ReadConstant, int WriteMultiplier, int WriteConstant)
Parameters:	Comm: Communications Port to which ServoCenter 4.1 controller is attached. ReadMultiplier: The multiplier used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is multiplied by the requested number of bytes to be read. ReadConstant: A constant used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is added to the product of the ReadMultiplier and the requested number of bytes. WriteMultiplier: The multiplier used to calculate the total time-out period for write operations, in milliseconds. For each write operation, this value is multiplied by the requested number of bytes to be written. WriteConstant: A constant used to calculate the total time-out period for read operations, in milliseconds. For each write operation, this value is added to the product of the WriteMultiplier and the bytes to be written.
Return Value:	0=Success, Non-0=Fail

3.2 Installing the yeisrvo.dll Runtime Library

For your programs to be able to use the ServoCenter 4.1 DLL, the DLL must first be placed somewhere that your running program will be able to find it. The best location for your DLL is in the same directory as your running program. For example, if your program is found in '`C:\ServoCenter\program\`', copy the ServoCenter 4.1 DLL to that directory.

Another location where you can store the ServoCenter 4.1 DLL is the folder where Windows stores the system-wide runtime libraries. In Windows 95 and 98, this folder is '`C:\WINDOWS\system\`.' In Windows ME, 2000, and XP, the folder is '`C:\WINDOWS\system32\`.'

Once the ServoCenter 4.1 DLL has been copied to one of these directories, you are ready to begin writing programs that use it.

3.3 Programming with yeisrvo.dll in Visual Basic 6.0

To use the functions provided by yeisrvo.dll from a Visual Basic program, you must first provide the Visual Basic environment with a way to know where to find these functions and how to treat them. To do this, the following form is used:

```
Public Declare Function InitPort Lib "yeisrvo.dll" ( _
    ByVal PortNum As Long, _
    ByVal Baud As Long _
) As Integer
```

This code snippet declares a public function named InitPort that is located in the yeisrvo.dll runtime library. The InitPort function receives two long integers as arguments and returns an integer.

To use any of the ServoCenter 4.1 DLL functions, a declaration like the one above must be written and stored in a code module. The ServoCenter.bas file, located on the ServoCenter 4.1 CD, contains declarations for all of the ServoCenter 4.1 runtime library's functions.

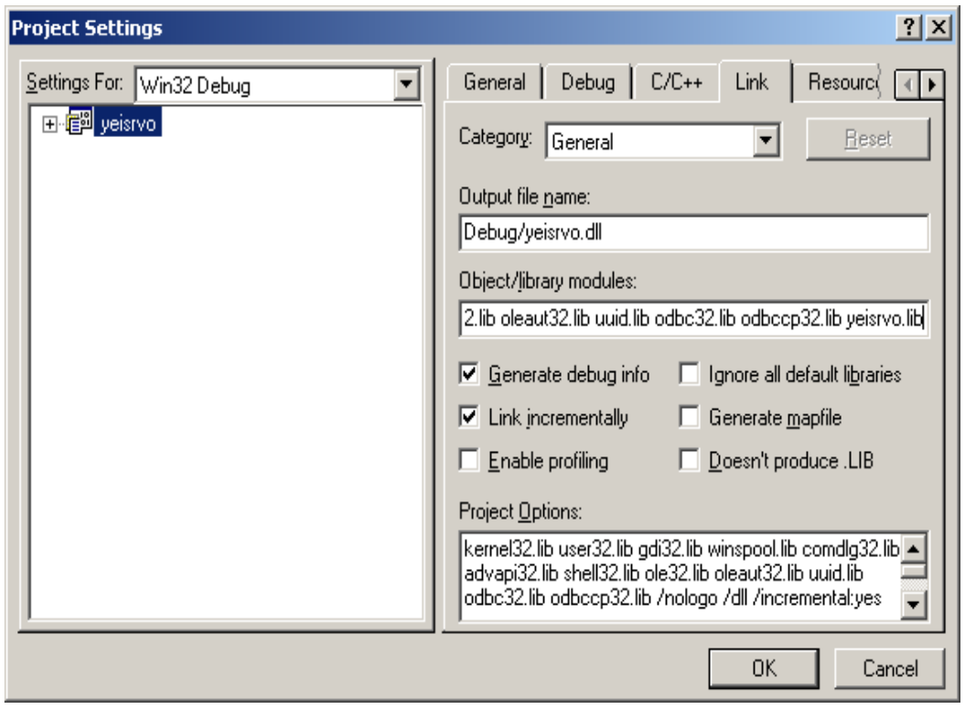
After making declarations for the DLL functions, they can be called in the same fashion as any other Visual Basic routine. The code snippet below illustrates calling a DLL function in Visual Basic.

```
Private Sub Form_Load()
    If InitPort(1, 9600) <> 0 Then
        MsgBox "Port could not be opened!"
    Else
        MsgBox "Port opened successfully!"
    End If
End Sub
```

This code snippet calls the InitPort() function declared previously. The ServoCenterDLLExample.vbp project, located on the ServoCenter 4.1 CD, is a sample Visual Basic project that demonstrates usage of the ServoCenter 4.1 DLL from the Visual Basic 6.0 environment.

3.4 Programming with yeisrvo.dll in Visual C++ 6.0

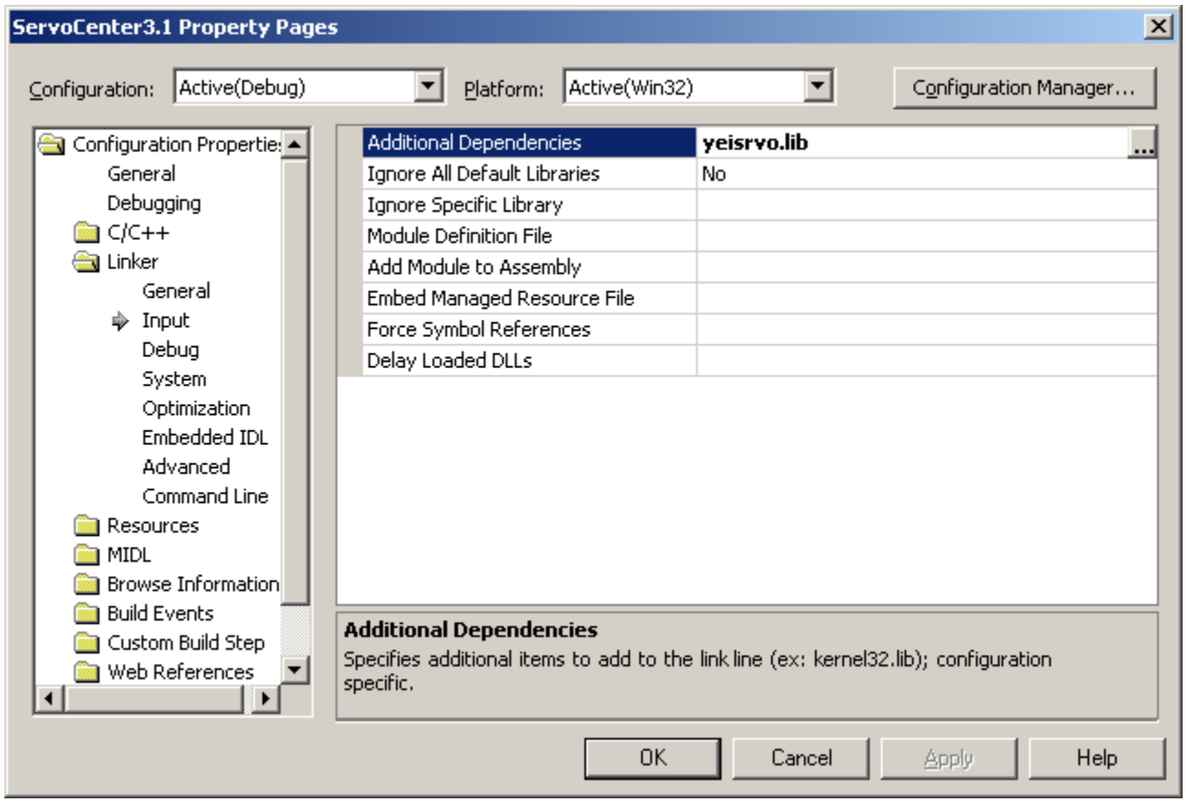
To use the ServoCenter 4.1 runtime library within the Visual C++ 6.0 environment, you must first copy the yeisrvo.lib import library to Visual C++ 6.0's library directory. By default, this is '`C:\Program Files\Microsoft Visual Studio\VC98\LIB.`' The yeisrvo.lib file contains object code that needs to be linked to your programs when they are compiled. To ensure that Visual C++ links to the import library, you will need to alter your project settings such that yeisrvo.lib is is in the link list. To do this, click **Project->Settings** and then click the **Link** tab. At the end of the list of **Object/library modules**, append a space and yeisrvo.lib. The following illustration shows a properly modified module list:



After adding yeisrvo.lib to the project's module list, you must now include the servocenter.h header file. This file contains function prototypes for the functions contained within the ServoCenter 4.1 runtime library. The file can be found on the ServoCenter 4.1 CD. After including the header file, you may call any of the functions listed above. The Visual C++ 6.0 project ServoCenterDLL.dsp, located on the ServoCenter 4.1 CD, contains examples of how to call these functions.

3.5 Programming with yeisrvo.dll in Visual C++ .NET

Programming with the yeisrvo.dll runtime library in Visual C++ .NET is very similar to programming in Visual C++ 6.0. The only real difference is the process by which the project settings are modified to use the yeisrvo.lib import library. For Visual C++ .NET, the default library directory is '`C:\Program Files\Microsoft Visual Studio .NET\Vc7\lib.`' To add yeisrvo.lib to the list of libraries to link, right-click on the name of the project in the Solutions Explorer, then select **Properties**. Select **Linker** from the **Property Pages** dialog that appears, and then click **Input**. Next add yeisrvo.lib to the **Additional Dependencies** and click **OK** to apply the settings. The illustration below shows a properly modified Properties Dialog:



Once the yeisrvo.lib import library has been added to your project, you may follow the instructions provided in Section 4.4.4 to program the ServoCenter 4.1 controller board. The Visual C++ .NET project, ServoCenterCppNETDll.vcproj, located on the ServoCenter 4.1 CD, contains examples of how to call these functions.

3.6 Programming with yeisrvo.dll in the Microsoft .NET Framework

The process for accessing the runtime library functions of the ServoCenter 4.1 DLL is very similar on both the Visual Basic .NET and the C# platforms. In both instances, the programs must import System.Runtime.InteropServices, which provides the DllImport function, and System.Text, which provides the StringBuilder class. The DllImport function allows you to call runtime library functions directly in your code, and the StringBuilder class provides a way to pass mutable strings to the DLL functions.

Visual Basic .NET

The following code snippets demonstrate importing classes and calling DLL functions in Visual Basic .NET.

```
Imports System.Runtime.InteropServices
Imports System.Text

<DllImport("yeisrvo.dll")> Function InitPort(ByVal PortNum As
Integer,   ByVal Baud As Long) As Integer
    End Function

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
          System.EventArgs) Handles MyBase.Load
        Call InitPort(1, 9600)
End Sub
```

The Visual Basic .NET project, ServoCenterVBNETDLL.vbproj, located on the ServoCenter 4.1 CD, demonstrates using the yeisrvo.dll functions from a VB.NET program.

C#

The following code snippets demonstrate importing classes and calling DLL functions in C#.

```
using System.Text;
using System.Runtime.InteropServices;

public class ServoCenter
{
     [DllImport("yeisrvo.dll",EntryPoint="InitPort")]
          public extern static int InitPort(int PortNum,long
BaudRate);
}

private void Form1_Load(object sender, System.EventArgs e)
{
     int i;
     if(ServoCenter.InitPort(1,9600)!=0)
     {
          Console.WriteLine("Could not open serial port!");
          return;
     }
}
```

The C# project, ServoCenterExampleC.csproj, located on the ServoCenter 4.1 CD, demonstrates using the yeisrvo.dll functions from a C# program.

4. Programming With the ServoCenter 4.1 .NET Assembly

The ServoCenter 4.1 .NET Assembly can be used in programs utilizing the .NET Framework to control the ServoCenter 4.1. This control is designed to allow the programmer to communicate with the ServoCenter 4.1 board without having to worry about the communications protocol. Below is an explanation of how to control the ServoCenter 4.1 using the ServoCenter 4.1 .NET Assembly.

4.1 Operation with ServoCenter 4.1 .NET Assembly

The ServoCenter 4.1 CD contains the ServoCenter 4.1 .NET Assembly. The ServoCenter Control was designed to handle the details of raw serial communication with the ServoCenter 4.1.

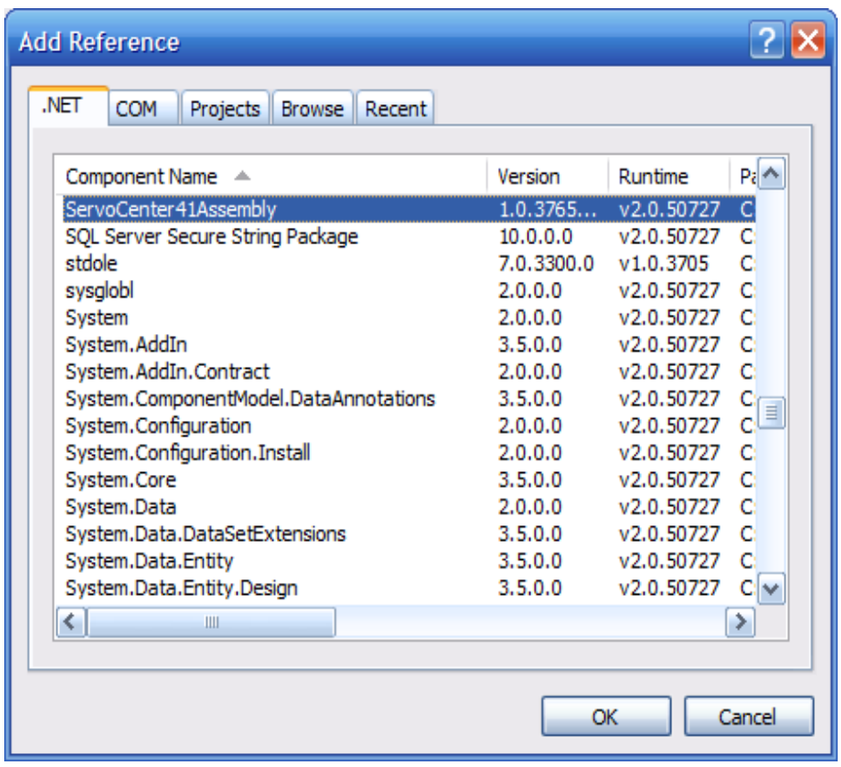
4.2 Installing the ServoCenter 4.1 .NET Assembly

Navigate to the “Resources” directory on the CD.

1. Run the ServoCenter 4.1 .NET Assembly Installer. To do this, double-click “ServoCenter 4.1 .NET Assembly Setup.exe” to begin the installation process.
2. Follow the onscreen instructions to complete the installation.
3. The control is now installed and ready for use!

4.3 Using the ServoCenter 4.1 .NET Assembly in Visual Basic 2008

1. Open a new project in the Visual Basic 2008 editor.
2. Add the ServoCenter41Assembly to your project. To do this, click the “Project” menu at the top of the editor and then select “Add Reference”. The menu will appear:



3. If the ServoCenter41Assembly is properly installed, it will appear in the dialog box. If not, it can still be added by clicking the “Browse” button and browsing to the directory where it is stored.

   Once the ServoCenter41Assembly has been selected, click the “OK” button to add it to your project.

4. With the ServoCenter41Assembly now added to your project, the contents of the assembly can be browsed through the Solution Explorer.

4.4 ServoCenter 4.1 .NET Assembly Methods

The functions provided by the ServoCenter 4.1 .NET Assembly correspond directly with the methods in the ServoCenter 4.1 DLL. For detailed information, please see section 3.1 of this document.

5. Appendix

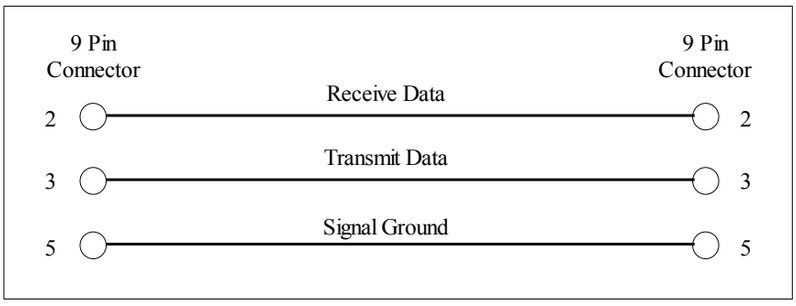
5.1 Hexadecimal/Decimal/Binary Conversion Chart

Decimal	Hex	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

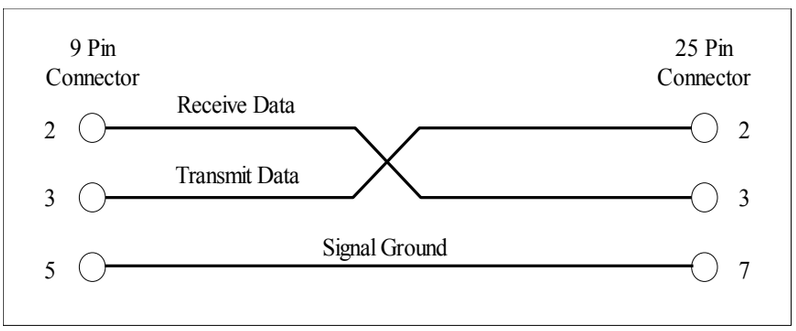
5.2 Serial Cable Diagram

The diagrams below illustrate the wire connections necessary for ServoCenter 4.1 compatible serial cables.

9-Pin to 9-Pin Serial Connection



9-Pin to 25-Pin Serial Connection



5.3 ServoCenter 4.1 Circuit Schematic